0828.68273 PATENT

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

In Re U.S. Patent Application)
)
Applicant: Shibayama et al.)
)
Serial No.)
)
Filed: August 26, 2003)
)
For: PARALLEL PROCESS)
EXECUTION METHOD AND)
MULTIPROCESSOR)
COMPUTER)
)
Art Unit:)

I hereby certify that this paper is being deposited with the United States Postal Service as EXPRESS MAIL in an envelope addressed to: Mail Stop Patent Application, Commissioner for Patents, P.O. Box 1450, Alexandria, VA 22313-1450, on this date.

8/26/2003 *Date*

Express Mail No. EL846225506US

CLAIM FOR PRIORITY

Commissioner for Patents
P.O. Box 1450
Alexandria, VA 22313-1450

Dear Sir:

Applicants claim priority benefits under 35 U.S.C. § 120 on the basis of the PCT application identified below:

PCT Patent Application No. PCT/JP01/01532 filed February 28, 2001

A certified copy of the earlier application is enclosed.

Respectfully submitted,

GREER, BURNS & CRAIN, LTD.

By
Patrick G. Burns
Registration No. 29,367

August 26, 2003
300 South Wacker Drive
Suite 2500
Chicago, Illinois 60606
Telephone: 312.360.0080
Facsimile: 312.360.9315

Patrick Burns
312-360-0080
1828.1601 US

日 本 国 特 許 庁

JAPAN PATENT OFFICE

別紙添付の書類は下記の出願書類の謄本に相違ないことを証明する。
This is to certify that the annexed is a true copy of the following application as filed with this Office.

出願年月日
Date of Application: ２００１年 ２月２８日

出願番号
Application Number: ＰＣＴ／ＪＰ０１／０１５３２

出願人
Applicant (s): 富士通株式会社
柴山 悟基
松嶋 雄典
菊嶋 薫

２００３ 年 ５ 月 ２０ 日

特許庁長官
Commissioner,
Japan Patent Office 太田信一郎

出証平 １５－５００１１２

受理官庁用写し

特許協力条約に基づく国際出願願書 FUP-1042P

原本（出願用）- 印刷日時 2001年02月26日 (26.02.2001) 月曜日 14時40分32秒

0	受理官庁記入欄	
0-1	国際出願番号.	PCT/JP01/01532
0-2	国際出願日	28.02.01
0-3	(受付印)	PCT International Application 日 本 国 特 許 庁
0-4	様式-PCT/RO/101 この特許協力条約に基づく国際出願願書は、	
0-4-1	右記によって作成された。	PCT-EASY Version 2.91 (updated 01.01.2001)
0-5	申立て 出願人は、この国際出願が特許協力条約に従って処理されることを請求する。	
0-6	出願人によって指定された受理官庁	日本国特許庁 (RO/JP)
0-7	出願人又は代理人の書類記号	FUP-1042P
I	発明の名称	並列プロセス実行方法、及びマルチプロセッサ型コンピュータ
II	出願人	
II-1	この欄に記載した者は	出願人である (applicant only)
II-2	右の指定国についての出願人である。	米国を除くすべての指定国 (all designated States except US)
II-4ja	名称	富士通株式会社
II-4en	Name	FUJITSU LIMITED
II-5ja	あて名:	211-8588 日本国 神奈川県 川崎市中原区上小田中４丁目 １番１号
II-5en	Address:	1-1, Kamikodanaka 4-chome, Nakahara-ku Kawasaki-shi, Kanagawa 211-8588 Japan
II-6	国籍（国名）	日本国 JP
II-7	住所（国名）	日本国 JP
II-8	電話番号	044-754-3034
II-9	ファクシミリ番号	044-754-3563

特許協力条約に基づく国際出願願書 FUP-1042P

原本（出願用）- 印刷日時 2001年02月26日 (26.02.2001) 月曜日 14時40分32秒

III-1	その他の出願人又は発明者	
III-1-1	この欄に記載した者は	出願人及び発明者である (applicant and inventor)
III-1-2	右の指定国についての出願人である。	米国のみ (US only)
III-1-4ja	氏名(姓名)	柴山 悟基
III-1-4en	Name (LAST, First)	SHIBAYAMA, Satoki
III-1-5ja	あて名:	211-8588 日本国 神奈川県 川崎市中原区上小田中４丁目１番１号 富士通株式会社内
III-1-5en	Address:	c/o FUJITSU LIMITED 1-1, Kamikodanaka 4-chome, Nakahara-ku Kawasaki-shi, Kanagawa 211-8588 Japan
III-1-6	国籍（国名）	日本国 JP
III-1-7	住所（国名）	日本国 JP
III-2	その他の出願人又は発明者	
III-2-1	この欄に記載した者は	出願人及び発明者である (applicant and inventor)
III-2-2	右の指定国についての出願人である。	米国のみ (US only)
III-2-4ja	氏名(姓名)	松嶋 雄典
III-2-4en	Name (LAST, First)	MATSUSHIMA, Yusuke
III-2-5ja	あて名:	211-8588 日本国 神奈川県 川崎市中原区上小田中４丁目１番１号 富士通株式会社内
III-2-5en	Address:	c/o FUJITSU LIMITED 1-1, Kamikodanaka 4-chome, Nakahara-ku Kawasaki-shi, Kanagawa 211-8588 Japan
III-2-6	国籍（国名）	日本国 JP
III-2-7	住所（国名）	日本国 JP

特許協力条約に基づく国際出願願書 FUP-1042P
原本（出願用）- 印刷日時 2001年02月26日 (26.02.2001) 月曜日 14時40分32秒

III-3	その他の出願人又は発明者	
III-3-1	この欄に記載した者は	出願人及び発明者である (applicant and inventor)
III-3-2	右の指定国についての出願人である。	米国のみ (US only)
III-3-4ja	氏名(姓名)	菊嶋 薫
III-3-4en	Name (LAST, First)	KIKUSHIMA, Kaoru
III-3-5ja	あて名:	211-8588 日本国 神奈川県 川崎市中原区上小田中４丁目 １番１号 富士通株式会社内
III-3-5en	Address:	c/o FUJITSU LIMITED 1-1, Kamikodanaka 4-chome, Nakahara-ku Kawasaki-shi, Kanagawa 211-8588 Japan
III-3-6	国籍（国名）	日本国 JP
III-3-7	住所（国名）	日本国 JP
IV-1	代理人又は共通の代表者、通知のあて名 下記の者は国際機関において右記のごとく出願人のために行動する。	代理人 (agent)
IV-1-1ja	氏名(姓名)	服部 毅巖
IV-1-1en	Name (LAST, First)	HATTORI, Kiyoshi
IV-1-2ja	あて名:	192-0082 日本国 東京都 八王子市東町 ９番８号 八王子東邦生命ビル 服部特許事務所
IV-1-2en	Address:	Hattori Patent Office Hachioji Tohoseimei Bldg. 9-8, Azuma-cho Hachioji-shi, Tokyo 192-0082 Japan
IV-1-3	電話番号	0426-45-6644
IV-1-4	ファクシミリ番号	0426-45-8578
V	国の指定	
V-1	広域特許 (他の種類の保護又は取扱いを求める場合には括弧内に記載する。)	EP: AT BE CH&LI CY DE DK ES FI FR GB GR IE IT LU MC NL PT SE TR 及びヨーロッパ特許条約と特許協力条約の締約国である他の国
V-2	国内特許 (他の種類の保護又は取扱いを求める場合には括弧内に記載する。)	JP US

特許協力条約に基づく国際出願願書　　FUP-1042P

原本（出願用）－印刷日時 2001年02月26日（26.02.2001）月曜日 14時40分32秒

V-5	指定の確認の宣言 出願人は、上記の指定に加えて、規則4.9(b)の規定に基づき、特許協力条約のもとで認められる他の全ての国の指定を行う。ただし、V-6欄に示した国の指定を除く。出願人は、これらの追加される指定が確認を条件としていること、並びに優先日から15月が経過する前にその確認がなされない指定は、この期間の経過時に、出願人によって取り下げられたものとみなされることを宣言する。		
V-6	指定の確認から除かれる国	なし（NONE）	
VI	優先権主張	なし（NONE）	
VII-1	特定された国際調査機関(ISA)	日本国特許庁（ISA/JP）	
VIII	照合欄	用紙の枚数	添付された電子データ
VIII-1	願書	5	-
VIII-2	明細書	25	-
VIII-3	請求の範囲	4	-
VIII-4	要約	1	fup1042p.txt
VIII-5	図面	18	-
VIII-7	合計	53	
	添付書類	添付	添付された電子データ
VIII-8	手数料計算用紙	✓	-
VIII-9	別個の記名押印された委任状		-
VIII-10	包括委任状の写し	✓	-
VIII-16	PCT-EASYディスク	-	フレキシブルディスク
VIII-17	その他	納付する手数料に相当する特許印紙を貼付した書面	-
VIII-17	その他	国際事務局の口座への振込を証明する書面	-
VIII-18	要約書とともに提示する図の番号	1	
VIII-19	国際出願の使用言語名：	日本語（Japanese）	
IX-1	提出者の記名押印		
IX-1-1	氏名(姓名)	服部 毅巖	

受理官庁記入欄

10-1	国際出願として提出された書類の実際の受理の日	28.02.01
10-2	図面：	
10-2-1	受理された	
10-2-2	不足図面がある	

特許協力条約に基づく国際出願願書 FUP-1042P

原本（出願用）－印刷日時 2001年02月26日（26.02.2001）月曜日 14時40分32秒

10-3	国際出願として提出された書類を補完する書類又は図面であってその後期間内に提出されたものの実際の受理の日（訂正日）	
10-4	特許協力条約第11条(2)に基づく必要な補完の期間内の受理の日	
10-5	出願人により特定された国際調査機関	ISA/JP
10-6	調査手数料未払いにつき、国際調査機関に調査用写しを送付していない	

国際事務局記入欄

11-1	記録原本の受理の日	

明　細　書

並列プロセス実行方法、及びマルチプロセッサ型コンピュータ

技術分野

本発明は並列プロセス実行方法、マルチプロセッサ型コンピュータ、並列プロセス実行プログラムおよびそのプログラムを記録した記録媒体に関し、特に並列プロセスと他のプロセスとを時分割で実行する並列プロセス実行方法、その方法を実行するマルチプロセッサ型コンピュータ、その方法をコンピュータに実行させる並列プロセス実行プログラムおよびそのプログラムを記録した記録媒体に関する。

背景技術

複数のプロセッサを搭載したコンピュータ（マルチプロセッサ型コンピュータ）では、１つのプログラムを複数のプロセッサで並列処理することが出来る。以下、並列処理可能なプログラムを並列プログラムと呼ぶ。並列プログラムを処理する場合、１つの並列プログラムから複数の並列プロセスが生成される。各並列プロセスは、互いに並行して実行可能なプロセスである。生成された複数の並列プロセスは、別々のプロセッサで並列に実行される。並列プロセスを実行する各プロセッサは互いにデータ交換をしながら、並列プログラムに定められた一連の処理を実行する。なお、ここでプロセスとは、１以上のスレッドを含む処理単位である。また、プロセッサとは、ＣＰＵ(Central Processing Unit)やＭＰＵ(Micro Processing Unit)などの処理装置のことである。以下の説明では、便宜上、プロセッサとしてＣＰＵを用いるものとする。

各並列プロセスの処理には、他の並列プロセスとの間でデータ交換（同期通信）を行うべきチェックポイントがある。データ交換が必要な２つの並列プロセスを実行する各ＣＰＵは、並列プロセスの処理がチェックポイントまで終了したら、データ交換処理を行う。一方の並列プロセスの処理が先にチェックポイントに達した場合には、その並列プロセスを実行するＣＰＵは、データ交換相手の並

列プロセスの処理がチェックポイントに達するまで同期待ちを行う。

ＣＰＵでのデータ処理には、同期待ち以外にもＩＯ(Input/Output)待ち等の待ち時間が発生する。待ち状態となったＣＰＵで他のプロセスを実行させれば、システム全体としての処理効率が向上する。そこで、ＣＰＵに時分割処理を行わせ、待ち状態のＣＰＵに他のプロセスの処理を実行させることが行われている。ここで他のプロセスは、並列プロセスであってもよいし、単一のＣＰＵで実行される非並列プロセスであってもよい。

なお、並列プロセスのデータ交換の同期待ちの間、ＣＰＵが他のプロセスを実行していると、データ交換相手の並列プロセスの処理がチェックポイントに達したときに、他のプロセスの処理が終了していないことがあり得る。すると、データ交換相手の並列プロセスを実行するＣＰＵに、同期待ちの時間が発生してしまう。このような同期待ち時間の発生は、コンピュータシステムの処理効率の低下を招く。

そこで、特開平１０－７４１５０号公報に記載されたプロセススケジューリング方法では、時分割処理を行うコンピュータシステムの各ＣＰＵに、並列プロセスおよび他のプロセスの動作開始・停止を、所定の期間単位（フェーズ）で同時に行わせている。すなわち、ある並列プログラムから生成された複数の並列プロセスは、各ＣＰＵにおいて同時に処理が開始され、同時に処理が停止される。これにより、並列プロセス実行時に発生する同期待ちの時間が、時分割処理を行わなかった場合と同じになる。その結果、並列プログラムを構成する並列プロセス間の同期待ちを最小限にし、システム効率の低下を防止することができる。

ところで、コンピュータで処理するプロセスには、ターンアラウンド時間（プロセスの実行開始から実行終了までの時間）の保証が必要なものがある。ターンアラウンド時間の保証が必要なものとして、たとえば、気象データの解析処理がある。膨大な量の気象データを解析する処理は、気象予報を発表する一定時間前に必ず終了している必要がある。

しかし、上記公報記載のプロセススケジューリング方法では、各フェーズの時間が固定であるため、各並列プログラム毎に、ターンアラウンド時間を保証することができない。たとえば、マルチプロセッサ型コンピュータの処理能力の５

０％を使用しないと、ターンアラウンド時間を保証できないような並列プログラムもあり得る。ところが、上記公報記載のプロセススケジューリング方法では、その並列プログラムの処理に１フェーズ分（たとえば１０％）の時間しか割り当てることができず、ターンアラウンド時間を保証できない。

発明の開示

本発明はこのような点に鑑みてなされたものであり、並列プロセス毎に、任意の割合でＣＰＵの処理時間を配分することができる並列プロセス実行方法、マルチプロセッサ型コンピュータ、並列プロセス実行プログラム、および並列プロセス実行プログラムを記録した記録媒体を提供することを目的とする。

本発明では上記課題を解決するために、図１に示すような並列プロセス実行方法が提供される。本発明の並列プロセス実行方法は、１つの並列プログラムＡから生成される複数の並列プロセスＡ１～Ａ４と他のプロセス（たとえば並列プログラムＢの並列プロセスＢ１～Ｂ４）とを、複数のプロセッサ１～４で時分割で処理する場合に適用される。

本発明では、所定の周期の中で、並列プログラムの処理に割り当てるべき時間配分率が設定される（ステップＳ１）。すると、並列プログラムに設定された時間配分率に従って、プロセスの切り替え処理が行われる（ステップＳ２）。すなわち、並列プログラムＡの各並列プロセスＡ１～Ａ４が複数のプロセッサ１～４中の１つに割り当てられ、複数のプロセッサ１～４において、割り当てられた並列プロセスＡ１～Ａ４の処理が同時に実行開始される。さらに、並列プロセスＡ１～Ａ４の実行が開始されてからの経過時間が、所定の周期内での並列プログラムＡに設定された時間配分率に応じた時間に達すると、複数のプロセッサ１～４において、割り当てられた並列プロセスの実行が同時に終了する。

これにより、並列プログラムＡに対して任意の時間配分率を設定すると、所定の時間中の時間配分率に応じた時間だけ、並列プログラムＡから生成された並列プロセスＡ１～Ａ４の処理に割り当てられる。しかも、並列プログラムＡから生成された並列プロセスＡ１～Ａ４の処理は、複数のプロセッサ１～４において同時に実行開始され、同時に実行終了される。その結果、マルチプロセッサ型コン

ピュータの処理能力のうち並列プログラムの処理に割り当てるべき割合を任意に設定することができ、ターンアラウンド時間の保証が可能となる。

本発明の上記および他の目的、特徴および利点は本発明の例として好ましい実施の形態を表す添付の図面と関連した以下の説明により明らかになるであろう。

図面の簡単な説明

図１は、本発明の原理構成図である。

図２は、本発明の実施の形態を適用したマルチプロセッサ型コンピュータのハードウェア構成例を示す図である。

図３は、本実施の形態を実現するためのＯＳの機能を示すブロック図である。

図４は、タイムスロット割り当てマップの例を示す図である。

図５は、タイムテーブルの例を示す図である。

図６は、プロセスの優先度を示す図である。

図７は、プロセスの切り替え状況を示すタイムチャートの例である。

図８は、タイムスロット毎のプロセス切り替え処理を示すフローチャートである。

図９は、図４に示したタイムスロット割り当てマップに従ったプロセス切り替え状況を示す図である。

図１０は、時間配分率を均等にした場合のプロセス切り替え例を示す図である。

図１１は、時間配分率を１対９にした場合のプロセス切り替え例を示す図である。

図１２は、時間配分率を２対８にした場合のプロセス切り替え例を示す図である。

図１３は、スループット優先ポリシーにおいて、並列プログラムと非並列プログラムとを時分割で処理する場合のプロセス切り替え例を示す図である。

図１４は、スループット優先ポリシーにおいて並列プログラム以外に処理すべきプログラムがない場合のプロセス切り替え例を示す図である。

図１５は、ターンアラウンド優先ポリシーの場合のプロセス切り替え例を示す図である。

図１６は、複数のノードで並列処理を行う場合のシステム構成例を示す図である。

図１７は、複数のノード間でのタイムスロット同期処理のフローチャートである。

図１８は、ノード間で同期処理が行われた場合の処理プロセスの切り替え状況の例を示す図である。

発明を実施するための最良の形態

以下、本発明の実施の形態を図面を参照して説明する。

図１は、本発明の原理構成図である。本発明に係る並列プロセス実行方法では、並列プログラムを、複数のプロセッサ１～４（各プロセッサの識別子は、＃０～＃３）を有するマルチプロセッサ型コンピュータにおいて時分割で処理させる。図１の例では、並列プログラムＡと並列プログラムＢとを時分割で処理する。

図１に示すように、まず、各並列プログラムＡ，Ｂに対して、時間配分率が設定される（ステップＳ１）。時間配分率とは、所定の周期（１サイクル）内で、各並列プログラムから生成されるプロセスがプロセッサを占有する時間の割合である。図１の例では、並列プログラムＡの時間配分率が３０％に設定されている。また、並列プログラムＢの時間配分率が７０％に設定されている。

次に、各並列プログラムＡ，Ｂに対して設定された時間配分率に従って、並列プロセスＡ１～Ａ４と並列プロセスＢ１～Ｂ４とのプロセス切り替え処理が行われる（ステップＳ２）。プロセス切り替え処理では、並列プロセスＡ１～Ａ４は、それぞれに対応するプロセッサにより並列に処理される。並列処理では、複数の並列プロセスＡ１～Ａ４の処理が同時に実行開始される。そして、並列プロセスＡ１～Ａ４の実行が開始されてからの経過時間が、所定の周期内での並列プログラムＡに設定された時間配分率（３０％）に応じた時間に達すると、複数のプロセッサ１～４において並列プロセスＡ１～Ａ４の実行が同時に終了する。

並列プロセスＡ１～Ａ４の実行が同時に終了すると、複数のプロセッサ１～４において並列プロセスＢ１～Ｂ４の処理が同時に実行開始される。そして、並列プロセスＢ１～Ｂ４の実行が開始されてからの経過時間が、所定の周期内での並

列プログラムＢに設定された時間配分率（７０％）に応じた時間に達すると、複数のプロセッサ１～４において並列プロセスＢ１～Ｂ４の実行が同時に終了する。

なお、１つの並列プログラムから生成された複数の並列プロセスを実行する過程では、適宜、並列プロセスを実行するプロセッサ同士で、互いにデータ交換が行われる。

このように、本発明では時分割処理を行うマルチプロセッサ型コンピュータを用いて、並列プログラムから生成された並列プロセスの処理に割り当てる時間配分率を、任意に設定することができる。そして、１サイクル中の時間配分率に応じた時間を、並列プロセスの処理に割り当てるようにしたため、ターンアラウンドを保証することが可能となる。しかも、１つの並列プログラムから生成された並列プロセスは、同時に実行開始し、同時に実行終了するようにしたため、時分割処理をしたことによるデータ交換のための同期待ちの時間の増加が防止される。

以下に、本発明の実施の形態について具体的に説明する。なお、以下の説明では、プロセッサとしてＣＰＵを用いるものとする。

図２は、本発明の実施の形態を適用したマルチプロセッサ型コンピュータのハードウェア構成例を示す図である。コンピュータ１０は、複数のＣＰＵ１１～１４によって装置全体が制御されている。各ＣＰＵ１１～１４は、バス１９を介して互いに接続されている。ＣＰＵ１１～１４は、ＲＡＭ２０に格納された共有プログラム等に基づいて生成されたプロセスを実行する。

各ＣＰＵ１１～１４には、バス１９を介して、ＲＡＭ２０、ハードディスク装置（ＨＤＤ）１５、グラフィック処理装置１６、入力インタフェース１７、および通信インタフェース１８が接続されている。

ＲＡＭ２０には、ＣＰＵ１１に実行させるＯＳ(Operating System)のプログラムの少なくとも一部や、並列プログラムの少なくとも一部が一時的に格納される。また、ＲＡＭ２０には、タイムスロット割り当てマップ等のデータが格納される。

ＨＤＤ１５は、ＯＳのプログラム、並列プログラム、および非並列プログラムなどが格納される。また、ＨＤＤ１５には、各種プログラムの実行に必要なデータが格納される。

グラフィック処理装置１６には、モニタ２１が接続されている。グラフィック処理装置１６は、ＣＰＵ１１～１４からの命令に従って、画像をモニタ２１の画面に表示させる。入力インタフェース１７には、キーボード２２とマウス２３とが接続されている。入力インタフェース１７は、キーボード２２やマウス２３から送られてくる信号を、バス１９を介してＣＰＵ１１～１４に送信する。

通信インタフェース１８は、ネットワーク２４に接続されている。ネットワーク２４は、たとえばインターネットのような広域ネットワークである。通信インタフェース１８は、ネットワーク２４を介して、他のコンピュータとの間でデータの送受信を行う。

以上のようなハードウェア構成によって、本実施の形態の処理機能を実現することができる。たとえば、図２に示したコンピュータの電源が投入されると、ＨＤＤ１５に格納されたＯＳのプログラムの一部が、ＲＡＭ２０に読み込まれる。そして、各ＣＰＵ１１～１４によりＯＳのプログラムが実行される。これにより、各ＣＰＵ１１～１４上でＯＳの動作が開始する。

図３は、本実施の形態を実現するためのＯＳの機能を示すブロック図である。本実施の形態では、コンピュータ１０内にＯＳ３０が立ち上げられる。

ＯＳ３０は、タイムスロット割り当てマップ３１、定義ファイル３２、タイムテーブル３３、ＧＵＩ３４、プログラム３５、プロセス実行部３６、プロセス切り替え部３７、およびコンテキスト３８を有している。プロセス実行部３６、プロセス切り替え部３７及びコンテキスト３８は、４つのＣＰＵ１１～１４それぞれに設けられている。ＯＳ３０上では、各ＣＰＵ１１～１４がそれぞれ識別子ＣＰＵ＃０、ＣＰＵ＃１、ＣＰＵ＃２、ＣＰＵ＃３で認識される。

タイムスロット割り当てマップ３１は、プロセスの実行タイミングの協調動作のスケジュールが定義されたデータである。タイムスロット割り当てマップ３１は、ユーザの操作入力により、任意のデータを登録することが出来る。タイムスロット割り当てマップ３１には、各ＣＰＵが実行すべきプロセスの割り当てが、タイムスロット毎に登録されている。このタイムスロット割り当てマップ３１をもとに、各ＣＰＵで現在のタイムスロットにどのプロセスが割り当てられているのかを知ることができる。タイムスロット割り当てマップ３１の詳細は後述する。

定義ファイル３２は、タイムスロット割り当てマップ３１においてプロセスが未割り当てのタイムスロットをどのように利用するのかのポリシー（判断基準）を示すスケジューリングポリシーの情報を含んでいる。スケジューリングポリシーには、たとえば、スループット優先ポリシーやターンアラウンド優先ポリシーがある。プロセス切り替えに適用するスケジューリングポリシーは、操作入力によって任意に選択することが出来る。

スループット優先ポリシーは、システム全体のスループット（単位時間当りの処理量）向上を優先させるためのポリシーである。スループット優先ポリシーでは、ＣＰＵがアイドル状態となる時間をできるだけ減らすように、プロセスが未割り当てのタイムスロット（空きタイムスロット）で何らかのプロセスを実行する。本実施の形態では、実行するプロセスの優先順は、対話型プロセス＞非並列プロセス＞並列プロセスの順とする。

ターンアラウンド優先ポリシーは、ターンアラウンドを保証するためのポリシーである。ターンアラウンド優先ポリシーでは、各プロセスに対して、割り当てられたＣＰＵ配分率以上にＣＰＵを使用させないようにする。すなわち、プロセスが未割り当てのタイムスロットでは、バッチプロセス（並列プロセスまたは非並列プロセス）を実行させない。

タイムテーブル３３は、１サイクル開始から、各タイムスロットが開始されるまでの時間が設定されたテーブル形式のデータである。タイムテーブル３３の詳細は後述する。

ＧＵＩ３４は、キーボード２２やマウス２３等の入力装置からの入力信号を解釈し、各種コマンドを生成する。ＧＵＩ３４は、生成した制御コマンドをプロセス実行部３６に送る。これにより、プロセス実行部３６において対話型プロセスが生成される。また、ＧＵＩ３４は、プロセス実行部３６から送られる情報を、モニタ２１に表示する。

複数のプログラム３５は、並列プログラムまたは非並列プログラムである。複数の並列プログラムは、ＣＰＵで並列処理が可能なプログラムである。非並列プログラムは、１つのＣＰＵで実行されるプログラムである。複数のプログラム３５は、それぞれの少なくとも一部が、たとえばＨＤＤ１５からＲＡＭ２０に読み

込まれる。

プロセス実行部３６は、複数のプログラム３５やＧＵＩ３４から送られるコマンドに基づいてプロセス（並列プロセスや非並列プロセス）を生成し、生成したプロセスを時分割で実行する。また、プロセス実行部３６は、実行中の並列プロセスの処理が、データ交換（同期通信）を行うべきチェックポイントに達すると、他のＣＰＵのプロセス実行部との間でデータ交換を行う。

プロセス切り替え部３７は、プロセス実行部３６で実行されるプロセスの切り替え制御を行う。具体的には、プロセス切り替え部３７は、タイムテーブル３３を参照し、タイムスロットの開始時刻を判断する。プロセス切り替え部３７は、タイムスロットの開始時刻になると、タイムスロット割り当てマップ３１を参照し、開始するタイムスロットで実行するプロセスを決定する。

タイムスロット割り当てマップ３１において、開始するタイムスロットにプロセスが割り当てられていない場合には、プロセス切り替え部３７は、定義ファイル３２を参照する。そして、プロセス切り替え部３７は、定義ファイル３２に含まれるスケジューリングポリシーに従って、開始するタイムスロットで実行するプロセスを決定する。

プロセス切り替え部３７は、開始するタイムスロットで実行するプロセスを決定したら、プロセス実行部３６が行っている処理を中断させ、コンテキストの切り替えを行う。すなわち、プロセス実行部３６が実行していたプロセスのコンテキストを待避させ、実行するプロセスに対応するコンテキストをプロセス実行部３６に渡す。

コンテキスト３８は、各プロセスを実行するのに必要な制御情報である。コンテキストには、プロセスが中断したときのプログラムカウンタの内容などが含まれている。

図４は、タイムスロット割り当てマップの例を示す図である。図４の例では、１サイクルが１０のタイムスロットに分割されている。各タイムスロットには、タイムスロット番号＃０～＃９が割り振られている。そして、タイムスロット割り当てマップ３１において、４つのＣＰＵ＃０～＃３それぞれに対して、各タイムスロット＃０～＃９で実行すべきプロセスが設定されている。

図４の例において、タイムスロット＃０～２では、全てのＣＰＵ＃０～＃３に対して、並列プログラムＡから生成された並列プロセスが設定されている。タイムスロット＃３，＃４では、ＣＰＵ＃０，＃１に対して、並列プログラムＢから生成された並列プロセスが設定されており、ＣＰＵ＃２，＃３に対して、並列プログラムＣから生成された並列プロセスが設定されている。タイムスロット＃５では、ＣＰＵ＃０，＃１に対して、並列プログラムＤから生成された並列プロセスが設定されており、ＣＰＵ＃２に対して、非並列プログラムＥから生成された非並列プロセスが設定されている。なお、タイムスロット＃５では、ＣＰＵ＃３にはプロセスが設定されておらず、空きタイムスロットとなっている。タイムスロット＃６～＃８では、全てのＣＰＵ＃０～＃３に対して、並列プログラムＦから生成された並列プロセスが設定されている。タイムスロット＃９では、全てのＣＰＵ＃０～＃３に対して、対話型プロセスＴが設定されている。なお、対話型プロセスに割り当てられたタイムスロットでは、ＯＳ上で生成される複数の対話型プロセスの処理が、優先順位の高い順に実行される。

図５は、タイムテーブルの例を示す図である。タイムテーブル３３には、タイムスロット番号毎に、オフセットタイムが設定されている。オフセットタイムとは、１つのサイクルが開始されてから各タイムスロットが開始されるまでの時間である。この例では、１サイクルは１msecであるものとする。

図５の例では、タイムスロット＃０のオフセットタイムは「+0msec」である。タイムスロット＃１のオフセットタイムは「+100msec」である。タイムスロット＃２のオフセットタイムは「+200msec」である。タイムスロット＃３のオフセットタイムは「+300msec」である。タイムスロット＃４のオフセットタイムは「+400msec」である。タイムスロット＃５のオフセットタイムは「+500msec」である。タイムスロット＃６のオフセットタイムは「+600msec」である。タイムスロット＃７のオフセットタイムは「+700msec」である。タイムスロット＃８のオフセットタイムは「+800msec」である。タイムスロット＃９のオフセットタイムは「+900msec」である。

各ＣＰＵにおいて、１サイクルが開始されてから各タイムスロットのオフセッ

トタイムに等しい時間経過すると、タイムスロット割り当てマップが参照される。そして、切り替え前のタイムスロットにおいて処理するプロセスと、タイムスロット切り替え後において処理するプロセスとが異なる場合には、プロセスの切り替えが行われる。プロセスの切り替えは、優先度を変更することによって行われる。

図６は、プロセスの優先度を示す図である。優先度は、たとえば６０段階程度に分かれている。この例では、最高優先度（Ｈ）と、最低優先度（Ｌ）とは、バッチプロセス用の優先度である。なお、本実施の形態では、対話型プロセス以外の非並列プロセスおよび並列プロセスは、バッチプロセスとして実行されるものとする。

最高優先度（Ｈ）より１ランク低い優先度（Ｈ－１）から、最低優先度（Ｌ）より２ランク高い優先度（Ｌ＋２）までは、対話型プロセス用の優先度である。最低優先度（Ｌ）より１ランク高い優先度（Ｌ＋１）は、ダミーアイドルプロセスおよび非並列プロセス用の優先度である。ここで、ダミーアイドルプロセスとは、並列プロセスを動作させないために用意されるプロセスである。ダミーアイドルプロセスを実行するＣＰＵは、無意味な処理を繰り返しており、実際には何も処理（計算等）をしない（実効が無い）。

各ＣＰＵは、タイムスロットの切り替えタイミングにおいて、次のタイムスロットで処理すべきプロセスがあれば、そのプロセスの優先度を、最高優先度（Ｈ）に設定する。また、各ＣＰＵは、処理対象外のバッチプロセスの優先度を、最低優先度（Ｌ）に設定する。

図６の例には、ＣＰＵ＃０のタイムスロット＃０～＃２における各プロセスの優先度を示している。プロセスＡ１（並列プログラムＡから生成された並列プロセス）には、最高優先度（Ｈ）が設定されている。対話型プロセスＴには、最低優先度（Ｌ）より２ランク上の優先度（Ｌ＋２）が設定されている。並列プロセスＢ１（並列プログラムＢから生成された並列プロセス）、並列プロセスＤ１（並列プログラムＤから生成された並列プロセス）および並列プロセスＦ１（並列プログラムＦから生成された並列プロセス）には、最低優先度（Ｌ）が設定されている。

時分割処理では、プロセスの切り替えタイミングにおいて、最も優先度の高いプロセスの処理が実行される。したがって、図６の例では、ＣＰＵ＃０においてプロセスＡ１が実行される。もし、全てのバッチプロセスに対して最低優先度（Ｌ）が設定されていれば、対話型プロセスＴが実行されることになる。

図７は、プロセスの切り替え状況を示すタイムチャートの例である。図７には、ＣＰＵ＃０のプロセスの切り替え状況の例を示している。

図７の例では、タイムスロット＃０の開始時点において、並列プロセスＡ１の優先度が最高優先度（Ｈ）に設定されている。他の並列プロセスの優先度は、最低優先度（Ｌ）である。また、対話型プロセスＴの優先度は、最高優先度（Ｈ）より１ランク低い優先度（Ｈ－１）から、最低優先度（Ｌ）より２ランク高い優先度（Ｌ＋２）の間の優先度が設定されている。その結果、タイムスロット＃０では、ＣＰＵ＃０において並列プロセスＡ１が実行される。

タイムスロット＃１の開始時点と、タイムスロット＃２との開始時点では、各プロセスの優先度が変化しない。そのため、タイムスロット＃１とタイムスロット＃２では、引き続き並列プロセスＡ１が実行される。これにより、１サイクル中の３０％の時間が、並列プロセスＡ１の処理に対して割り当てられる。

タイムスロット＃３の開始時点では、並列プロセスＡ１に対して、最低優先度（Ｌ）が設定され、並列プロセスＢ１に対して、最高優先度（Ｈ）が設定される。他の並列プロセスの優先度は変化しない。これにより、タイムスロット＃３では、ＣＰＵ＃０において並列プロセスＢ１が実行される。タイムスロット＃４の開始時点では、各プロセスの優先度が変化しない。そのため、タイムスロット＃４では、引き続き並列プロセスＢ１が実行される。これにより、１サイクル中の２０％の時間が、並列プロセスＢ１の処理に対して割り当てられる。

タイムスロット＃５の開始時点では、並列プロセスＢ１に対して、最低優先度（Ｌ）が設定され、並列プロセスＤ１に対して、最高優先度（Ｈ）が設定される。他の並列プロセスの優先度は変化しない。これにより、タイムスロット＃５では、ＣＰＵ＃０において並列プロセスＤ１が実行される。これにより、１サイクル中の１０％の時間が、並列プロセスＤ１の処理に対して割り当てられる。

タイムスロット＃６の開始時点では、並列プロセスＤ１に対して、最低優先度

（Ｌ）が設定され、並列プロセスＦ１に対して、最高優先度（Ｈ）が設定される。これにより、タイムスロット＃６では、ＣＰＵ＃０において並列プロセスＦ１が実行される。タイムスロット＃７の開始時点と、タイムスロット＃８との開始時点では、各プロセスの優先度が変化しない。そのため、タイムスロット＃７とタイムスロット＃８では、引き続き並列プロセスＦ１が実行される。これにより、１サイクル中の３０％の時間が、並列プロセスＦ１の処理に対して割り当てられる。

タイムスロット＃９の開始時点では、並列プロセスＦ１に対して、最低優先度（Ｌ）が設定される。他の並列プロセスの優先度は変化しない。これにより、タイムスロット＃９では、対話型プロセスＴの優先度が最も高くなり、ＣＰＵ＃０において対話型プロセスTが実行される。これにより、１サイクル中の１０％の時間が、全ての対話型プロセスＴの処理に対して割り当てられる。

図８は、タイムスロット毎のプロセス切り替え処理を示すフローチャートである。この処理は、各タイムスロットの開始時刻になったときに、ＯＳを実行する各ＣＰＵにおいて実行される処理である。以下に、図８の処理をステップ番号に沿って説明する。

［ステップＳ１１］プロセス切り替え部３７（図３に示す）は、定義ファイル３２（図３に示す）からスケジューリングポリシーの設定を読み出す。

［ステップＳ１２］プロセス切り替え部３７は、スケジューリングポリシーが、ターンアラウンド優先ポリシーか否かを判断する。ターンアラウンド優先ポリシーであれば、処理がステップＳ１３に進められる。ターンアラウンドポリシーでなければ、処理がステップＳ１４に進められる。

［ステップＳ１３］プロセス切り替え部３７は、ダミーアイドルプロセスを生成し、ダミーアイドルプロセスに対して最低優先度より１レベル高い優先度（Ｌ＋１）を設定する。

［ステップＳ１４］プロセス切り替え部３７は、各ＣＰＵに対して同時に割り込みを発生させる。

［ステップＳ１５］プロセス切り替え部３７は、タイムスロットを１つ進める。

［ステップＳ１６］プロセス切り替え部３７は、タイムスロット割り当てマッ

プ３１（図３に示す）を参照し、現タイムスロットが空きか否かを判断する。現タイムスロットが空きであれば、処理がステップＳ１９に進められる。現タイムスロットが空きでなければ、処理がステップＳ１７に進められる。

［ステップＳ１７］プロセス切り替え部３７は、タイムスロット割り当てマップ３１を参照し、現タイムスロットの割り当てが、対話型プロセスか否かを判断する。対話型プロセスが割り当てられてられていれば、処理がステップＳ２２に進められる。対話型プロセスが割り当てられていなければ、処理がステップＳ１８に進められる。

［ステップＳ１８］プロセス切り替え部３７は、現タイムスロットに割り当てられたプロセスの優先度を最高値に設定する。その後、処理がステップＳ２２に進められる。

［ステップＳ１９］ステップＳ１６において、現タイムスロットが空きと判断された場合には、プロセス切り替え部３７は、定義ファイル３２を参照し、スループット優先ポリシーか否かを判断する。スループット優先ポリシーの場合には、処理がステップＳ２０に進められる。スループット優先ポリシーではない場合には、処理がステップＳ２２に進められる。

［ステップＳ２０］プロセス切り替え部３７は、非並列プロセスが存在するか否かを判断する。非並列プロセスが存在する場合には、処理がステップＳ２１に進められる。非並列プロセスが存在しない場合には、処理がステップＳ２２に進められる。

［ステップＳ２１］プロセス切り替え部３７は、非並列プロセスに対して、最低優先度よりも１レベル高い優先度（Ｌ＋１）を設定する。

［ステップＳ２２］プロセス切り替え部３７は、優先度が最高値の並列プロセスのうち、現タイムスロットに割り当てられたプロセス以外の並列プロセスの優先度を最低値（Ｌ）にする。

［ステップＳ２３］プロセス切り替え部３７は、タイムテーブル３３を参照し、次のプロセッサ割り込みの時刻を設定する。その後、処理がステップＳ１４に進められる。

このようにして、タイムスロット毎に各プロセスの優先度を変えることができ

る。その結果、タイムスロット割り当てマップおよびスケジューリングポリシーに従ったプロセス切り替えが行われる。

ここで、スケジューリングポリシーがターンアラウンド優先ポリシーの場合、ダミーアイドルプロセス（優先度Ｌ＋１）が生成され（ステップＳ１３）、空きタイムスロットにおいてバッチプロセスの優先度が最低値（Ｌ）とされる（ステップＳ２２）。そのため、対話型プロセスが存在すればその対話型プロセスが実行され、対話型プロセスが存在しなければ、ダミーアイドルプロセスが実行される。これにより、空きタイムスロットにおいて実効あるバッチプロセスが実行されることはなくなる。

また、スケジューリングポリシーがスループット優先ポリシーの場合、空きタイムスロットにおいて、非並列プロセスの優先度が最低値より１ランク上（Ｌ＋１）に設定され（ステップＳ２１）、並列プロセスの優先度が最低値（Ｌ）とされる（ステップＳ２２）。そのため、対話型プロセス｛優先度（Ｌ＋２）～（Ｈ－１）｝が存在すればその対話型プロセスが実行され、対話型プロセスが存在しなければ、存在する非並列プロセスが実行される。これにより、対話型プロセス、非並列プロセス、並列プロセスの順番で優先的に実行される。

図９は、図４に示したタイムスロット割り当てマップに従ったプロセス切り替え状況を示す図である。

図９に示すように、タイムスロット＃０～タイムスロット＃２の期間（１サイクルの３０％の時間）、ＣＰＵ＃０において並列プロセスＡ１が実行され、ＣＰＵ＃１において並列プロセスＡ２が実行され、ＣＰＵ＃２において並列プロセスＡ３が実行され、ＣＰＵ＃３において並列プロセスＡ４が実行される。なお、並列プロセスＡ１～Ａ４は、共に並列プログラムＡから生成されたバッチプロセスである。

タイムスロット＃３～タイムスロット＃４の期間（１サイクルの２０％の時間）、ＣＰＵ＃０において並列プロセスＢ１が実行され、ＣＰＵ＃１において並列プロセスＢ２が実行され、ＣＰＵ＃２において並列プロセスＣ１が実行され、ＣＰＵ＃３において並列プロセスＣ２が実行される。なお、並列プロセスＢ１，Ｂ２は、共に並列プログラムＢから生成されたバッチプロセスである。また、並

列プロセスＣ１，Ｃ２は、共に並列プログラムＣから生成されたバッチプロセスである。

タイムスロット＃５の期間（１サイクルの１０％の時間）、ＣＰＵ＃０において並列プロセスＤ１が実行され、ＣＰＵ＃１において並列プロセスＤ２が実行され、ＣＰＵ＃２において非並列プロセスＥ１が実行され、ＣＰＵ＃３においてスケジューリングポリシーに従って決定されたプロセスＸが実行される。なお、並列プロセスＤ１，Ｄ２は、共に並列プログラムＤから生成されたバッチプロセスである。また、非並列プロセスＥ１は、非並列プログラムＥから生成されたバッチプロセスである。

タイムスロット＃６～タイムスロット＃８の期間（１サイクルの３０％の時間）、ＣＰＵ＃０において並列プロセスＦ１が実行され、ＣＰＵ＃１において並列プロセスＦ２が実行され、ＣＰＵ＃２において並列プロセスＦ３が実行され、ＣＰＵ＃３において並列プロセスＦ４が実行される。なお、並列プロセスＦ１～Ｆ４は、共に並列プログラムＦから生成されたバッチプロセスである。

タイムスロット＃９の期間（１サイクルの１０％の時間）、各ＣＰＵ＃０～＃３それぞれにおいて、対話型プロセスＴが実行される。

以上のように、ＣＰＵ毎に、各プロセスに割り振る時間配分率を任意に指定できることにより、顧客の要望に応じた時間配分率で、並列プログラムを処理することが出来る。以下に、プロセスに対する時間配分率の設定例を説明する。

以下の例では、並列プログラムＡは、４つのＣＰＵを常に占有して並列処理を行った場合、処理を終了するのに１日を要するものとする。また、並列プログラムＢは、４つのＣＰＵを常に占有して並列処理を行った場合、処理を終了するのに４日を要するものとする。

図１０は、時間配分率を均等にした場合のプロセス切り替え例を示す図である。図１０の例では、並列プログラムＡに対して５０％の配分率（タイムスロット＃０～＃４）を割り当て、並列プログラムＢに対して５０％（タイムスロット＃５～＃９）の配分率を割り当てている。これにより、タイムスロット＃０～＃４の期間、並列プログラムＡから生成される並列プロセスＡ１～Ａ４が各ＣＰＵ＃０～＃３で実行される。また、タイムスロット＃５～＃９の期間、並列プログラム

Ｂから生成される並列プロセスＢ１～Ｂ４が各ＣＰＵ＃０～＃３で実行される。

図１０の場合には、並列プログラムＡと並列プログラムＢとの処理を同時に実行開始すると、処理の所要時間の短い並列プログラムＡの処理が先に終了する。並列プログラムＡの処理終了後は、並列プログラムＢの処理が１００％の配分率で行われる。すなわち、並列プログラムＡの処理は、処理開始から２日後に終了する。また、並列プログラムＢの処理は、処理開始から５日後に終了する。

図１１は、時間配分率を１対９にした場合のプロセス切り替え例を示す図である。図１１の例では、並列プログラムＡに対して１０％の配分率（タイムスロット＃０）を割り当て、並列プログラムＢに対して９０％（タイムスロット＃１～＃９）の配分率を割り当てている。これにより、タイムスロット＃０の期間、並列プログラムＡから生成される並列プロセスＡ１～Ａ４が各ＣＰＵ＃０～＃３で実行される。また、タイムスロット＃１～＃９の期間、並列プログラムＢから生成される並列プロセスＢ１～Ｂ４が各ＣＰＵ＃０～＃３で実行される。

図１１の場合には、並列プログラムＡと並列プログラムＢとの処理を同時に実行開始すると、処理の所要時間の長い並列プログラムＢの処理が先に終了する。並列プログラムＡの処理終了後は、並列プログラムＢの処理が１００％の配分率で行われる。すなわち、並列プログラムＡの処理は、処理開始から５日後に終了する。また、並列プログラムＢの処理は、処理開始から４．４日後に終了する。

このように、並列プログラムに対するＣＰＵの処理時間の配分率を任意に設定することで、処理の所要時間の長い並列プログラムＢを先に終了させることが可能となる。

図１２は、時間配分率を２対８にした場合のプロセス切り替え例を示す図である。図１２の例では、並列プログラムＡに対して２０％の配分率（タイムスロット＃０～＃１）を割り当て、並列プログラムＢに対して８０％（タイムスロット＃２～＃９）の配分率を割り当てている。これにより、タイムスロット＃０～＃１の期間、並列プログラムＡから生成される並列プロセスＡ１～Ａ４が各ＣＰＵ＃０～＃３で実行される。また、タイムスロット＃２～＃９の期間、並列プログラムＢから生成される並列プロセスＢ１～Ｂ４が各ＣＰＵ＃０～＃３で実行される。

図１２の場合には、並列プログラムＡと並列プログラムＢとの処理を同時に実行開始すると、並列プログラムＡと並列プログラムＢとの処理がほぼ同時に終了する。すなわち、並列プログラムＡと並列プログラムＢとの処理は、共に処理開始から５日後に終了する。

このように、並列プログラムに対するＣＰＵの処理時間の配分率を任意に設定することで、処理の所要時間の異なる複数の並列プログラムの処理を、同時に終了させることが可能となる。

次に、スケジューリングポリシーの違いに応じたプロセス切り替えの違いについて説明する。

図１３は、スループット優先ポリシーにおいて、並列プログラムと非並列プログラムとを時分割で処理する場合のプロセス切り替え例を示す図である。

ここで、各ＣＰＵ＃０～＃３に対して、タイムスロット＃０～＃４の期間、並列プログラムＡの処理を行わせるものとする。ＣＰＵ＃０～＃２に対して、タイムスロット＃５～＃９の期間、並列プログラムＢの処理を行わせるものとする。そして、ＣＰＵ＃３に対するタイムスロット＃５～＃９の期間は、空きが設定されているものとする。また、並列プログラムＡと並列プログラムＢ以外に処理すべきプログラムとして非並列プログラムＥがあり、対話型プロセスは生成されていないものとする。

図１３の例では、タイムスロット＃０～＃４の期間、並列プログラムＡから生成される並列プロセスＡ１～Ａ４が各ＣＰＵ＃０～＃３で実行される。また、タイムスロット＃５～＃９の期間、並列プログラムＢから生成される並列プロセスＢ１～Ｂ３がＣＰＵ＃０～＃２で実行される。さらに、タイムスロット＃５～＃９の期間、並列プログラムＥから生成される非並列プロセスＥ１がＣＰＵ＃３で実行される。

このように、スループット優先ポリシーを選択し、空きタイムスロットにおいて非並列プロセスを実行すれば、システム全体の効率（スループット）がよくなる。

図１４は、スループット優先ポリシーにおいて並列プログラム以外に処理すべきプログラムがない場合のプロセス切り替え例を示す図である。

ここで、各ＣＰＵ＃０～＃３に対して、タイムスロット＃０～＃４の期間、並列プログラムＡの処理を行わせるものとする。ＣＰＵ＃０～＃２に対して、タイムスロット＃５～＃９の期間、並列プログラムＢの処理を行わせるものとする。そして、ＣＰＵ＃３に対するタイムスロット＃５～＃９の期間は、空きが設定されているものとする。また、並列プログラムＡと並列プログラムＢ以外に、処理すべきプログラム、あるいは対話型プロセスがないものとする。

図１４の例では、タイムスロット＃０～＃４の期間、並列プログラムＡから生成される並列プロセスＡ１～Ａ４が各ＣＰＵ＃０～＃３で実行される。また、タイムスロット＃５～＃９の期間、並列プログラムＢから生成される並列プロセスＢ１～Ｂ３が各ＣＰＵ＃０～＃２で実行される。

さらに、タイムスロット＃５～＃９の期間、並列プログラムＡから生成される並列プロセスＡ４がＣＰＵ＃３で実行される。しかし、ＣＰＵ＃３でタイムスロット＃５～＃９の期間、並列プロセスＡ４の処理実行中に、他の並列プロセスとの間でデータ交換（同期通信）を行うべきチェックポイントに達すると、ＣＰＵ＃３は同期待ちの状態となる。そして、次のサイクルで、他のＣＰＵ＃０～＃２において、並列プロセスＡ１～Ａ２の処理が進められることで、データ交換が可能となり、同期待ちの状態が解除される。

このように、スループット優先ポリシーにおいて、空きタイムスロットで実行すべき非並列プロセスあるいは対話型プロセスがなければ、並列プロセスの処理が進められる。これにより、システム全体の効率（スループット）はよくなる。ただし、並列プロセスの同期待ちの時間は、その並列プロセスの処理中の時間として換算される。すなわち、サーバコンピュータの処理機能を顧客に使用させ、ＣＰＵの動作時間に応じて料金を徴収していた場合、同期待ちのＣＰＵループの時間も課金されてしまう。すると、同じ並列プログラムを実行した場合であっても、空きのタイムスロットがあるか否かにより料金が異なってしまう。

そこで、顧客が、並列プログラムの内容が同じであれば同じ料金が徴収されることを望む場合には、ターンアラウンド優先ポリシーが選択される。また、顧客によりターンアラウンドの保証が望まれる場合にも、ターンアラウンド優先ポリシーが選択される。

図１５は、ターンアラウンド優先ポリシーの場合のプロセス切り替え例を示す図である。

ここで、各ＣＰＵ＃０～＃３に対して、タイムスロット＃０～＃４の期間、並列プログラムＡの処理を行わせるものとする。ＣＰＵ＃０～＃２に対して、タイムスロット＃５～＃９の期間、並列プログラムＢの処理を行わせるものとする。そして、ＣＰＵ＃３に対するタイムスロット＃５～＃９の期間は、空きが設定されているものとする。

図１５の例では、タイムスロット＃０～＃４の期間、並列プログラムＡから生成される並列プロセスＡ１～Ａ４が各ＣＰＵ＃０～＃３で実行される。また、タイムスロット＃５～＃９の期間、並列プログラムＢから生成される並列プロセスＢ１～Ｂ３が各ＣＰＵ＃０～＃２で実行される。さらに、タイムスロット＃５～＃９の期間、ダミーアイドルプロセスＧがＣＰＵ＃３で実行される。

ダミーアイドルプロセスＧが実行されている時間は、並列プロセスの処理時間として換算されない。従って、並列プログラムが各ＣＰＵを占有する時間の総計は、空きのタイムスロットの有無に依存しない。その結果、並列プログラムの内容が同じであれば同じ料金が徴収される。

次に、複数のコンピュータ（以下、ノードという）で並列処理を行う場合の例について説明する。

図１６は、複数のノードで並列処理を行う場合のシステム構成例を示す図である。図１６の例では、協働制御装置１００に対して、割り込み通知装置２０１、３０１を介して、複数のノード２００，３００が接続されている。ノード２００は、複数のＣＰＵ２１１，２１２を有している。同様に、ノード３００は、複数のＣＰＵ３１１,３１２を有している。

協働制御装置１００は、各ノード間での協働動作を制御する。たとえば、協働制御装置１００は、各ノード間で交換されるデータを中継する。また、協働制御装置１００は、システムの起動時に、各ノードに対して同期用の割り込み信号を通知する。

割り込み通知装置２０１，３０１は、協働制御装置１００からの割り込み信号を受け取り、ノード内の所定のＣＰＵに対して、割り込み信号を通知する。たと

えば、ノード２００に接続された割り込み通知装置２０１は、ＣＰＵ２１１に対して割り込み信号を通知する。ノード３００に接続された割り込み通知装置３０１は、ＣＰＵ３１１に対して割り込み信号を通知する。

各ノード２００，３００では、任意のＣＰＵ２１１，３１１が割り込み信号の通知を受け取ると、タイムスロットの初期化やノード内の各ＣＰＵ割り込みの設定が行われる。

図１７は、複数のノード間でのタイムスロット同期処理のフローチャートである。以下に、図１７の処理をステップ番号に沿って説明する。なお、図１７の例では、ノード２００に接続された割り込み通知装置２０１を用いて、割り込みタイマの設定を行うものとする。

［ステップＳ３１］ノード２００は、割り込み通知装置２０１に対して、並列処理の開始時刻を通知する。具体的には、ユーザからの操作入力に応答して、ノード２００で実行されているＯＳから割り込み通知装置２０１に対して開始時刻が通知される。

［ステップＳ３２］割り込み通知装置２０１は、協働制御装置１００に対して開始時刻を通知する。これにより、協働制御装置１００において、開始時刻を割り込み発生時刻としたタイマが設定される。

［ステップＳ３３］協働制御装置１００は、タイマ設定に基づいて、開始時刻に達したか否かを判断する。開始時刻に達していれば、処理がステップＳ３４に進められる。開始時刻に達していなければステップＳ３３の処理が繰り返される。

［ステップＳ３４］協働制御装置１００は、タイマに設定された開始時刻に達すると、協働制御装置１００に接続された全ての割り込み通知装置２０１，３０１に対してタイマ割り込みを通知する。

［ステップＳ３５］タイマ割り込みを受け取った各割り込み通知装置２０１，３０１は、それぞれに対応するノードの所定のＣＰＵに対して、タイマ割り込みを通知する。

［ステップＳ３６］タイマ割り込みの通知を受けた各ＣＰＵは、あらかじめ用意されている割り込みハンドラの処理ルーチン（プログラム）を起動する。すると、割り込みハンドラの処理に従って、タイムスロットの初期化（現行のタイム

スロット番号を＃０にする処理等）と、ＣＰＵ割り込み設定（所定の時刻にノード内の全てのＣＰＵに対して同時に発生させる割り込み信号の設定）が行われる。ＣＰＵ割り込みの設定は、全てのノードの全てのＣＰＵに対して、同時刻に割り込みが発生するように設定される。

このように、全てのＣＰＵに対する同時刻の割り込み設定が行われることにより、各ノードの各ＣＰＵにおいて、同時に割り込みが発生する。そして、タイムスロットの切り替え処理のサイクルが開始される。

図１８は、ノード間で同期処理が行われた場合の処理プロセスの切り替え状況の例を示す図である。図１８の例では、ノード＃０とノード＃１とを同時に起動した場合の、各ＣＰＵにおける処理を示している。ノード＃０とノード＃１とが同時に起動されると、それぞれのＣＰＵにおいて、起動処理が実行される。起動処理としては、たとえば、セルフテストの実施や、所定のブートプログラムのロード等の処理がある。起動処理の終了時刻は、ＣＰＵによって異なる。起動処理が終了したＣＰＵは、ＣＰＵの割り込み発生を待つ。

全てのＣＰＵにおいて起動処理が終了した後、全てのＣＰＵに対して同時に、割り込みを発生させる。各ＣＰＵは、割り込みの発生後、タイムスロットの切り替え処理を開始し、タイムスロット毎に、タイムスロット割り当てマップに設定されたプロセスを実行する。

このように、複数のノードにおいて、タイムスロットの切り替え処理のタイミングを同期させることにより、並列プログラムが生成される複数の並列プロセスを、複数のノードで分散して並列処理することが可能となる。すなわち、タイムスロットの切り替え処理のタイミングを同期させない場合、並列プログラムＡの各並列プロセスが別々のノードで異なる時間帯に実行される場合が発生し得る。この場合、異なるノードで処理されている複数の並列プロセス間において、データ交換のための同期待ち時間が過大になる虞がある。

ところが、タイムスロットの切り替え処理のタイミングを同期させることで、複数のノードで分散して処理される各並列プロセスが同時に実行開始され、同時に実行が終了される。その結果、データ交換時の同期待ち時間を最小限に抑えることが可能となる。

以上説明したように、本発明の実施の形態によれば、並列プログラムを処理する際のＣＰＵの時間配分率を任意に設定できることで、ターンアラウンド時間を保証することが可能となる。ターンアラウンド時間を保証することで、定期的に同じプログラムを実行するような場合に、他のプログラムの動作の有無に関係なく、いつも同じ時間で処理を終了させることができる。処理の終了時間の予測ができれば、処理結果を利用した作業を、予定通りに進めることができる。たとえば、気象予報のデータ処理において、システムの処理負荷に関係なく、常に予報の発表時刻よりも一定時間前に、気象データの解析を終了させることができる。

また、並列プログラムを処理する際のＣＰＵの時間配分率を任意に設定できることは、バッチプロセスごとにランク分けが可能となることを意味する。すなわち、重要なバッチプロセスには、多くのＣＰＵ配分率を与える。これにより、たとえば、重要でないバッチプロセスよりも早く重要なバッチプロセスの処理を終了させることができる。また、他のバッチプロセスよりも処理の所要時間が長いバッチプロセスの処理を、他のバッチプロセスの処理と同時に終了させることもできる。

また、対話型プロセス用のタイムスロットを設定出来るようにしたことにより、一定時間間隔で、確実に対話型プロセスを実行させることができる。すなわち、全てのタイムスロットに対してバッチプロセス（たとえば並列プロセス）を割り当てると、バッチプロセスが終了するまで、各タイムスロットにおいてバッチプロセスが最優先で実行される。すると、バッチプロセスが終了するまで、対話型プロセスが実行出来なくなってしまう。そこで、本実施の形態では、対話型プロセス用のタイムスロットにおいては、全てのバッチプロセスの優先度を、最低の値で設定するようにした。これにより、対話型プロセス全体へのＣＰＵの時間配分率が保証できる。

また、本実施の形態では、スケジューリングポリシーによって、スループット優先ポリシーとターンアラウンド優先ポリシーとのどちらかを、任意に選択できるようにした。これにより、顧客のニーズに応じて、スループットを優先したり、ターンアラウンドを優先したりすることができる。すなわち、本実施の形態では、プロセスが割当てられていないタイムスロットで、並列プロセスを動作させるか

否かを、システム単位で指定することが可能となった。

たとえば、課金を気にせずに、頻繁に同期をとらない並列プロセスが主に実行されるようなシステムではスループット優先ポリシーを選択し、何も割当てられでいないタイムスロットでいずれかのプロセスを動作させる。これにより、システム全体でのスループットが向上する。しかも、空きタイムスロットで実行するプロセスを選択する際の優先順位を、対話型プロセス、非並列プロセス、並列プロセスの順にしたため、空きタイムスロットで実行される処理において同期待ちが発生する可能性がさらに低くなる。

一方、頻繁に同期をとる並列プロセスが主に実行されるようなシステムでは、何も割当てられでいない空きタイムスロットで並列プロセスを実行しても、ＣＰＵの処理能力が無駄に消費されるだけである。この場合、空きタイムスロットで並列プロセスを実行しても、システム全体のスループットが向上しないばかりか、課金にもブレが生じてしまう。このような場合は、ターンアラウンド優先ポリシーに設定することで、何も割り当てられていないタイムスロットでは並列プロセスを実行しないようにすることで、ＣＰＵの処理能力が無駄に消費されるのを防止できる。

さらに、複数のＣＰＵが複数のノードに分散している場合、各ノードのＣＰＵで実行されるタイムスロットのサイクルを同期させるようにした。これにより、１つの並列プログラムから生成される複数の並列プロセスを、複数のノードで並列に処理しても、１つのノード内で並列処理したときと同程度の同期待ちしか発生しない。すなわち、複数のノードで並列処理することによる処理効率の低下を防止することが出来る。

なお、上記の例では、マルチＣＰＵにおける並列処理において、プロセスの実行タイミングをスケジュールに合わせて協調動作させるように説明したが、１つのノード内であれば、複数のスレッド間での協調動作をスケジューリングすることもできる。すなわち、タイムスロットの割り当てマップにおいて、スレッド単位で各ＣＰＵの処理対象を割り当てることも可能である。

なお、上記の処理内容は、コンピュータで読み取り可能な記録媒体に記録されたプログラムに記述されており、このプログラムをコンピュータで実行すること

により、上記処理がコンピュータで実現される。コンピュータで読み取り可能な記録媒体としては、磁気記録装置や半導体メモリ等がある。市場へ流通させる場合には、ＣＤ－ＲＯＭ(Compact Disk Read Only Memory)やフロッピーディスク等の可搬型記録媒体にプログラムを格納して流通させることができる。また、ネットワークを介して接続されたコンピュータの記憶装置にプログラムを格納しておき、ネットワークを介して接続された他のコンピュータにプログラムを転送することもできる。コンピュータで実行する際には、コンピュータ内のハードディスク装置等にプログラムを格納しておき、メインメモリにロードして実行することができる。

以上説明したように本発明では、並列プログラムに対して任意の時間配分率が設定されると、並列プロセスの処理を複数のプロセッサにおいて同時に実行開始し、所定の時間中の時間配分率に応じた時間経過後に、並列プロセスの処理を同時に実行終了するようにした。これにより、並列プログラムを処理する際のプロセッサの処理時間の時間配分率を任意に設定し、ターンアラウンド時間を保証することが可能となる。

上記については単に本発明の原理を示すものである。さらに、多数の変形、変更が当業者にとって可能であり、本発明は上記に示し、説明した正確な構成および応用例に限定されるものではなく、対応するすべての変形例および均等物は、添付の請求項およびその均等物による本発明の範囲とみなされる。

請 求 の 範 囲

1． 1つの並列プログラムから生成される複数の並列プロセスと他のプロセスとを、複数のプロセッサで時分割で処理するための並列プロセス実行方法において、

所定の周期の中で前記並列プログラムの処理に割り当てるべき時間配分率を設定し、

前記並列プログラムの前記各並列プロセスを前記複数のプロセッサ中の1つに割り当て、前記複数のプロセッサにおいて、割り当てられた並列プロセスの処理を同時に実行開始し、

前記並列プロセスの実行が開始されてからの経過時間が、前記所定の周期内での前記並列プログラムに設定された前記時間配分率に応じた時間に達すると、前記複数のプロセッサにおいて、割り当てられた並列プロセスの実行を同時に終了する、

ことを特徴とする並列プロセス実行方法。

2． 前記所定の周期を複数のタイムスロットに分割し、各プロセッサにおいて各タイムスロットで実行すべきプロセスを設定することにより、前記並列プログラムの処理に割り当てるべき時間配分率を設定することを特徴とする請求項1記載の並列プロセス実行方法。

3． 前記各タイムスロットで実行すべきプロセスには、対話型プロセスが含まれることを特徴とする請求項2記載の並列プロセス実行方法。

4． プロセスが設定されていない空きタイムスロットにおける処理対象の判断基準があらかじめ定義されており、前記各プロセッサは、前記空きタイムスロットにおいては、前記判断基準に従って、処理対象となるプロセスを選択することを特徴とする請求項2記載の並列プロセス実行方法。

5． 前記判断基準には、前記空きタイムスロットにおいて、バッチプロセスを実行可能なスループット優先ポリシーと、前記空きタイムスロットにおいて、バッチプロセスを実行しないターンアラウンド優先ポリシーとがあらかじめ用意されており、前記スループット優先ポリシーと前記ターンアラウンド優先ポリシー

とから任意に選択できることを特徴とする請求項４記載の並列プロセス実行方法。

６．　前記スループット優先ポリシーでは、プロセス選択の優先順位の高い方から、対話型プロセス、単一のプロセッサで実行される非並列プロセス、並列プロセスの順番であることを特徴とする請求項５記載の並列プロセス実行方法。

７．　前記複数のプロセッサが複数のノードに分散して配置されている場合には、各ノードに対して同時に割り込みを発生させ、当該割り込みを前記各ノードが有する各プロセッサに同時に通知し、割り込み発生のタイミングに合わせて前記各プロセッサによる前記所定の周期の計時を開始することを特徴とする請求項１記載の並列プロセス実行方法。

８．　１つの並列プログラムから生成される複数の並列プロセスと他のプロセスとを時分割で処理するための複数のプロセッサを有するマルチプロセッサ型コンピュータにおいて、

所定の周期の中で前記並列プログラムの処理に割り当てるべき時間配分率を設定する時間配分率設定手段と、

前記並列プログラムの前記各並列プロセスを前記複数のプロセッサ中の１つに割り当て、前記複数のプロセッサにおいて、割り当てられた並列プロセスの処理を同時に実行開始させ、前記並列プロセスの実行が開始されてからの経過時間が、前記所定の周期内での前記並列プログラムに設定された前記時間配分率に応じた時間に達すると、前記複数のプロセッサにおいて、それぞれに割り当てられた並列プロセスの実行を同時に終了させるプロセス実行手段と、

を有することを特徴とするマルチプロセッサ型コンピュータ。

９．　前記時間配分率設定手段では、前記所定の周期を複数のタイムスロットに分割し、各プロセッサにおいて各タイムスロットで実行すべきプロセスを設定することを特徴とする請求項８記載のマルチプロセッサ型コンピュータ。

１０．　プロセスが設定されていない空きタイムスロットにおける各プロセッサの処理対象の判断基準があらかじめ定義されており、

前記プロセス実行手段は、前記各プロセッサに対して、前記空きタイムスロットにおいては、前記判断基準に従って処理対象となるプロセスを選択させることを特徴とする請求項９記載のマルチプロセッサ型コンピュータ。

１１．　前記判断基準には、前記空きタイムスロットにおいて、バッチプロセスを実行可能なスループット優先ポリシーと、前記タイムスロットにおいて、バッチプロセスを実行しないターンアラウンド優先ポリシーとがあらかじめ用意されており、前記スループット優先ポリシーと前記ターンアラウンド優先ポリシーとから任意に選択できることを特徴とする請求項１０記載のマルチプロセッサ型コンピュータ。

１２．　１つの並列プログラムから生成される複数の並列プロセスと他のプロセスとを時分割で処理するための複数のプロセッサが、複数のノードに分散して配置されたマルチプロセッサ型コンピュータにおいて、

前記複数のノードそれぞれに対して、割り込み通知を同時に発生させる協働制御装置と、

所定の周期の中で前記並列プログラムの処理に割り当てるべき時間配分率を設定する時間配分率設定手段と、

前記並列プログラムの前記各並列プロセスを前記複数のプロセッサ中の１つに割り当て、前記複数のプロセッサにおいて、割り当てられた並列プロセスの処理を同時に実行開始させ、前記並列プロセスの実行が開始されてからの経過時間が、前記所定の周期内での前記並列プログラムに設定された前記時間配分率に応じた時間に達すると、前記複数のプロセッサにおいて、それぞれに割り当てられた並列プロセスの実行を同時に終了させる、前記複数のノードそれぞれに設けられた複数のプロセス実行手段と、

を有することを特徴とするマルチプロセッサ型コンピュータ。

１３．　１つの並列プログラムから生成される複数の並列プロセスと他のプロセスとを、複数のプロセッサで時分割で処理するための並列プロセス実行プログラムにおいて、

コンピュータに、

所定の周期の中で前記並列プログラムの処理に割り当てるべき時間配分率を設定し、

前記並列プログラムの前記各並列プロセスを前記複数のプロセッサ中の１つに割り当て、前記複数のプロセッサにおいて、割り当てられた並列プロセスの処理

を同時に実行開始し、

前記並列プロセスの実行が開始されてからの経過時間が、前記所定の周期内での前記並列プログラムに設定された前記時間配分率に応じた時間に達すると、前記複数のプロセッサにおいて、それぞれに割り当てられた並列プロセスの実行を同時に終了する、

処理を実行させることを特徴とする並列プロセス実行プログラム。

１４．　１つの並列プログラムから生成される複数の並列プロセスと他のプロセスとを、複数のプロセッサで時分割で処理するためのプログラムを記録したコンピュータ読み取り可能な記録媒体において、

前記コンピュータに、

所定の周期の中で、前記並列プログラムの処理に割り当てるべき時間配分率を設定し、

前記並列プログラムの前記各並列プロセスを前記複数のプロセッサ中の１つに割り当て、前記複数のプロセッサにおいて、割り当てられた並列プロセスの処理を同時に実行開始し、

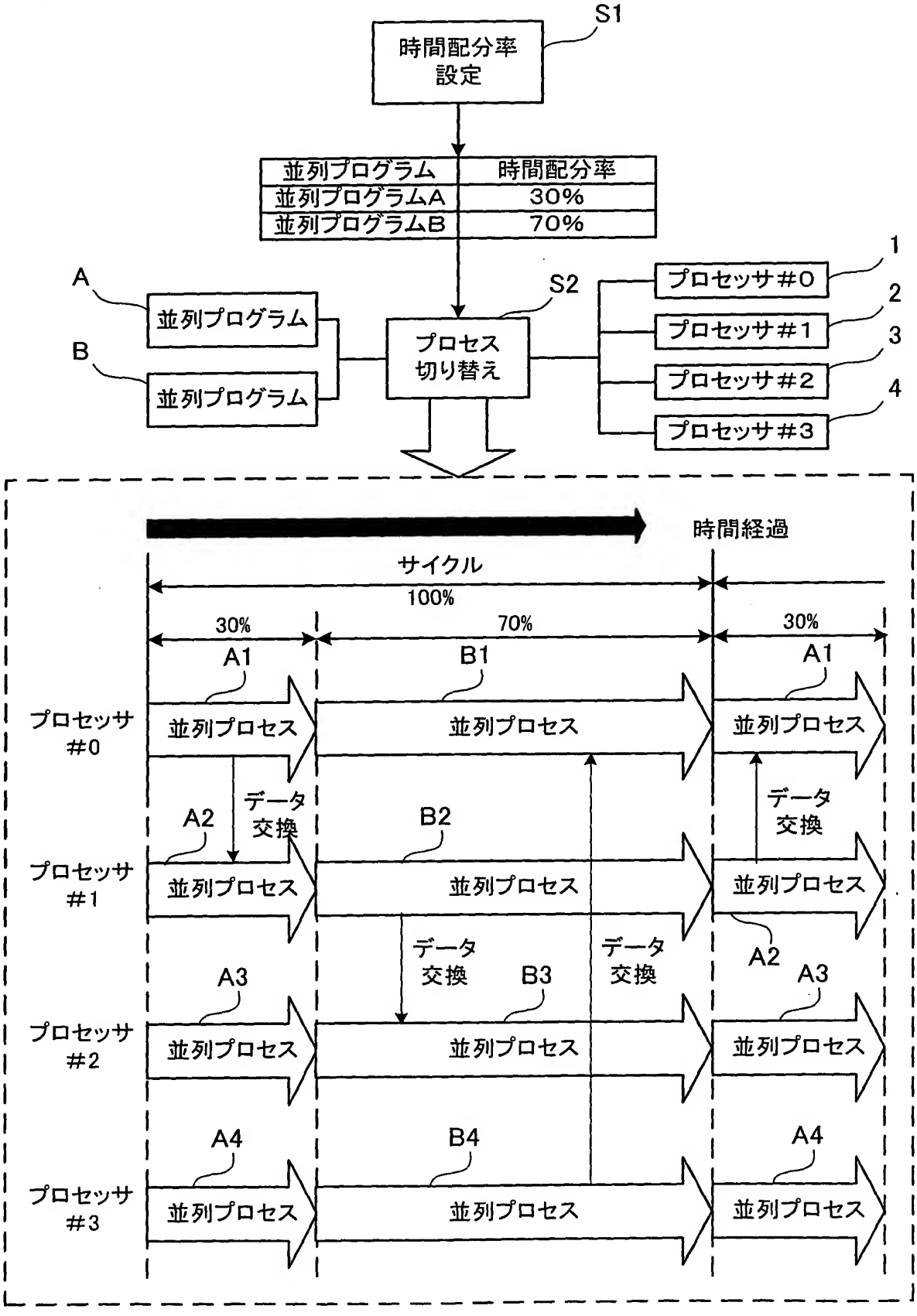
前記並列プロセスの実行が開始されてからの経過時間が、前記所定の周期内での前記並列プログラムに設定された前記時間配分率に応じた時間に達すると、前記複数のプロセッサにおいて、それぞれに割り当てられた並列プロセスの実行を同時に終了する、

処理を実行させることを特徴とする記録媒体。

要　約　書

並列プロセス毎に、任意の割合でＣＰＵの処理時間を配分することができるようにする。並列プログラムの処理に割り当てるべき時間配分率が設定される（ステップＳ１）。すると、並列プログラムに設定された時間配分率に従って、プロセスの切り替え処理が行われる（ステップＳ２）。すなわち、並列プログラム（Ａ）の各並列プロセス（Ａ１）～（Ａ４）が複数のプロセッサ（１）～（４）中の１つに割り当てられ、複数のプロセッサ（１）～（４）において、割り当てられた並列プロセス（Ａ１）～（Ａ４）の処理が同時に実行開始される。さらに、並列プロセス（Ａ１）～（Ａ４）の実行が開始されてからの経過時間が、所定の周期内での並列プログラム（Ａ）に設定された時間配分率に応じた時間に達すると、複数のプロセッサ（１）～（４）において、割り当てられた並列プロセスの実行が同時に終了する。

図1


S1
時間配分率
設定
並列プログラム
時間配分率
並列プログラムA
30%
並列プログラムB
70%
A
B
並列プログラム
並列プログラム
S2
プロセス
切り替え
1
プロセッサ#0
2
プロセッサ#1
3
プロセッサ#2
4
プロセッサ#3
時間経過
サイクル
100%
30%
70%
30%
A1
B1
A1
プロセッサ
#0
並列プロセス
並列プロセス
並列プロセス
データ
交換
A2
B2
データ
交換
プロセッサ
#1
並列プロセス
並列プロセス
並列プロセス
A2
データ
交換
B3
データ
交換
A3
A3
プロセッサ
#2
並列プロセス
並列プロセス
並列プロセス
A4
B4
A4
プロセッサ
#3
並列プロセス
並列プロセス
並列プロセス

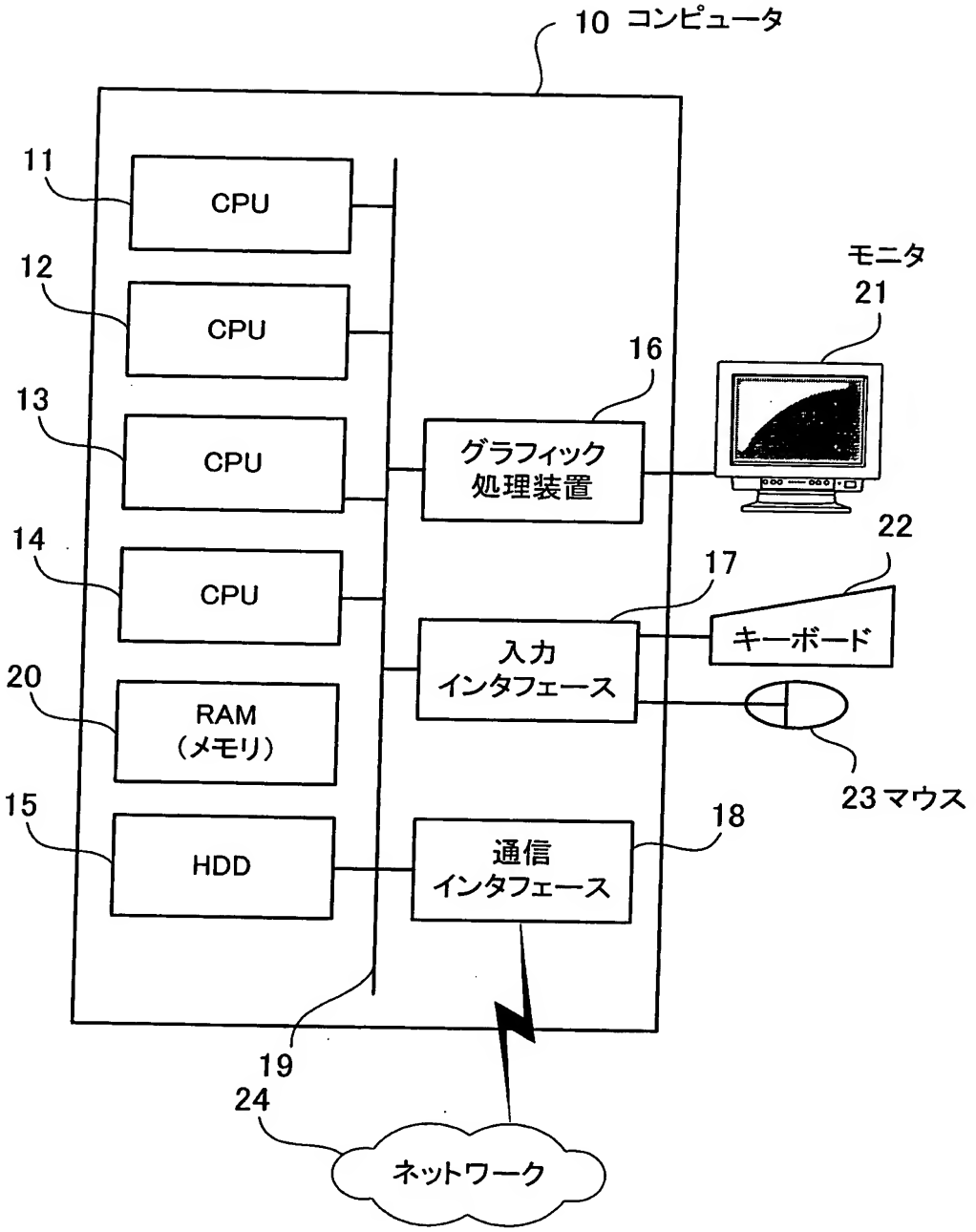

10 コンピュータ
11
CPU
12
CPU
13
CPU
14
CPU
20
RAM
(メモリ)
15
HDD
16
グラフィック
処理装置
17
入力
インタフェース
18
通信
インタフェース
19
モニタ
21
22
キーボード
23 マウス
24
ネットワーク

図2

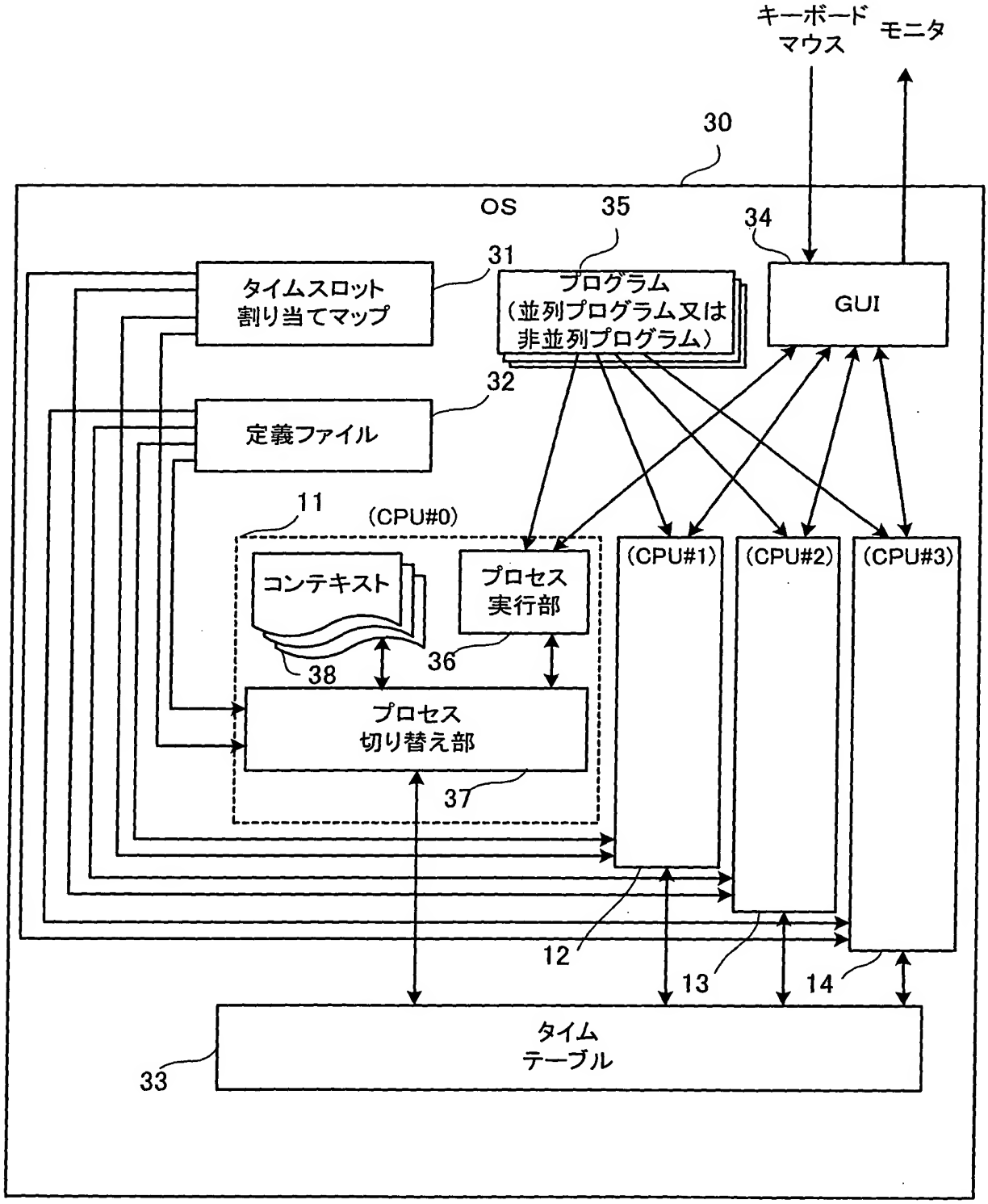

キーボード
マウス
モニタ
30
OS
35
34
31
タイムスロット
割り当てマップ
プログラム
(並列プログラム又は
非並列プログラム)
GUI
32
定義ファイル
11
(CPU#0)
コンテキスト
プロセス
実行部
(CPU#1)
(CPU#2)
(CPU#3)
38
36
プロセス
切り替え部
37
12
13
14
タイム
テーブル
33

図3

31 タイムスロット割り当てマップ

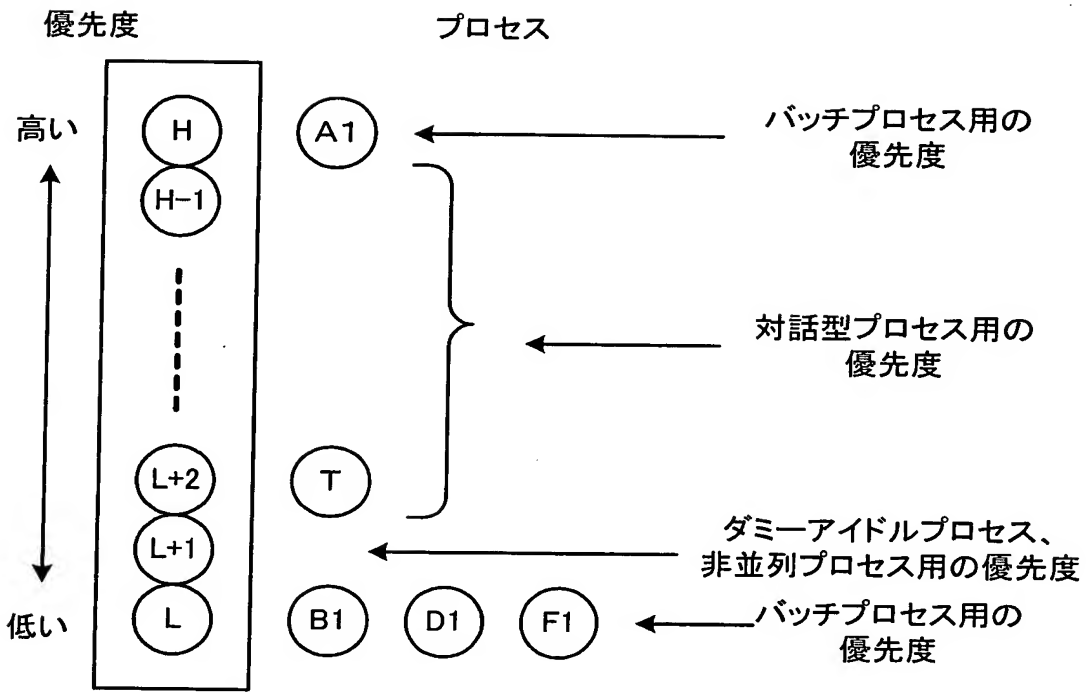
	タイムスロット番号									
	#0	#1	#2	#3	#4	#5	#6	#7	#8	#9
CPU#0	A	A	A	B	B	D	F	F	F	T
CPU#1	A	A	A	B	B	D	F	F	F	T
CPU#2	A	A	A	C	C	E	F	F	F	T
CPU#3	A	A	A	C	C	空	F	F	F	T

図4

33 タイムテーブル

タイムスロット番号	0	1	2	3	4	5	6	7	8	9
オフセットタイム格納領域	+0msec	+100msec	+200msec	+300msec	+400msec	+500msec	+600msec	+700msec	+800msec	+900msec

図5



CPU#0、タイムスロット#1～#2の
プロセスの優先度

図6

図7

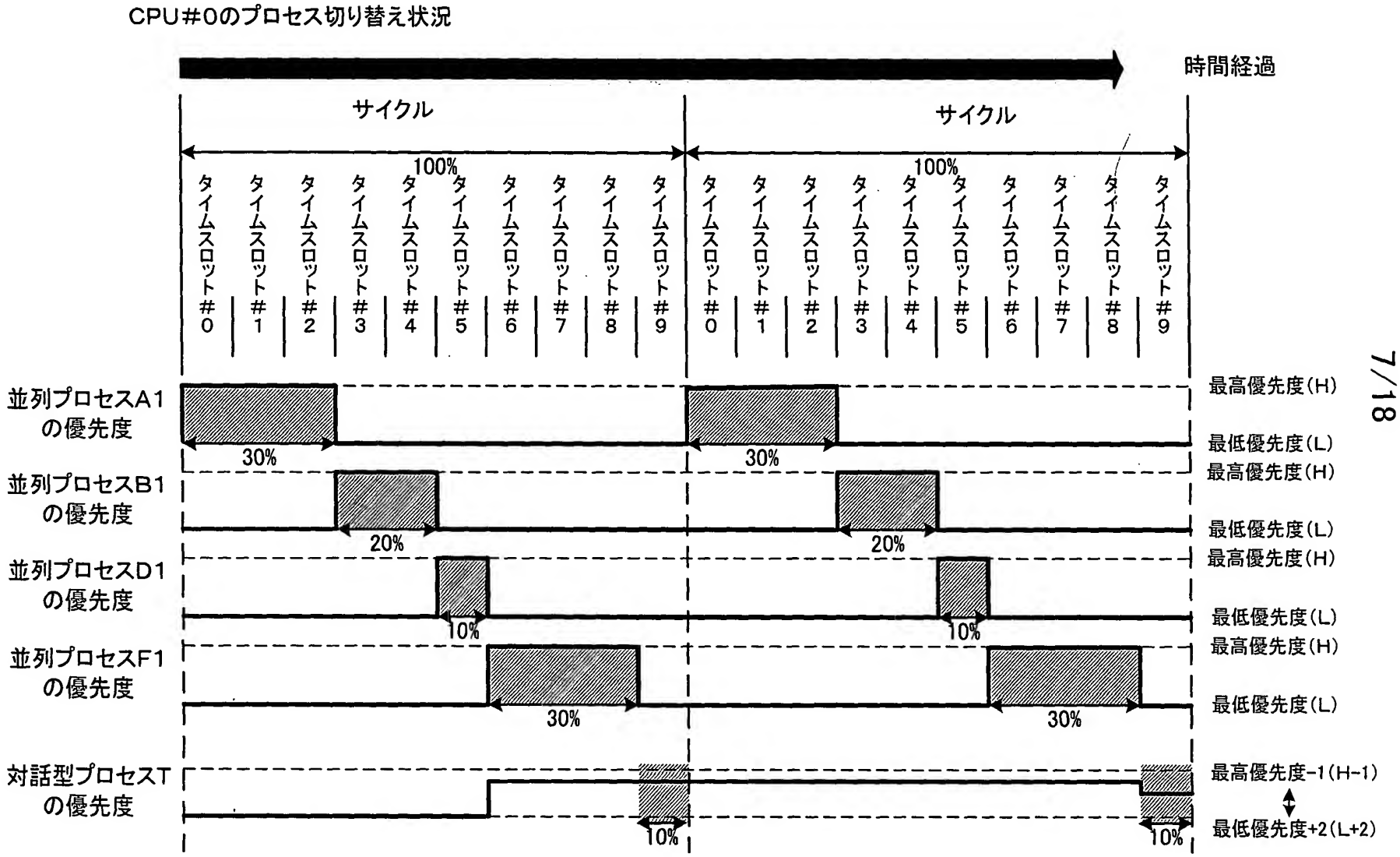

CPU#0のプロセス切り替え状況
時間経過
サイクル
サイクル
100%
100%
タイムスロット#0
タイムスロット#1
タイムスロット#2
タイムスロット#3
タイムスロット#4
タイムスロット#5
タイムスロット#6
タイムスロット#7
タイムスロット#8
タイムスロット#9
並列プロセスA1の優先度
並列プロセスB1の優先度
並列プロセスD1の優先度
並列プロセスF1の優先度
対話型プロセスTの優先度
30%
20%
10%
最高優先度(H)
最低優先度(L)
最高優先度-1(H-1)
最低優先度+2(L+2)

図8

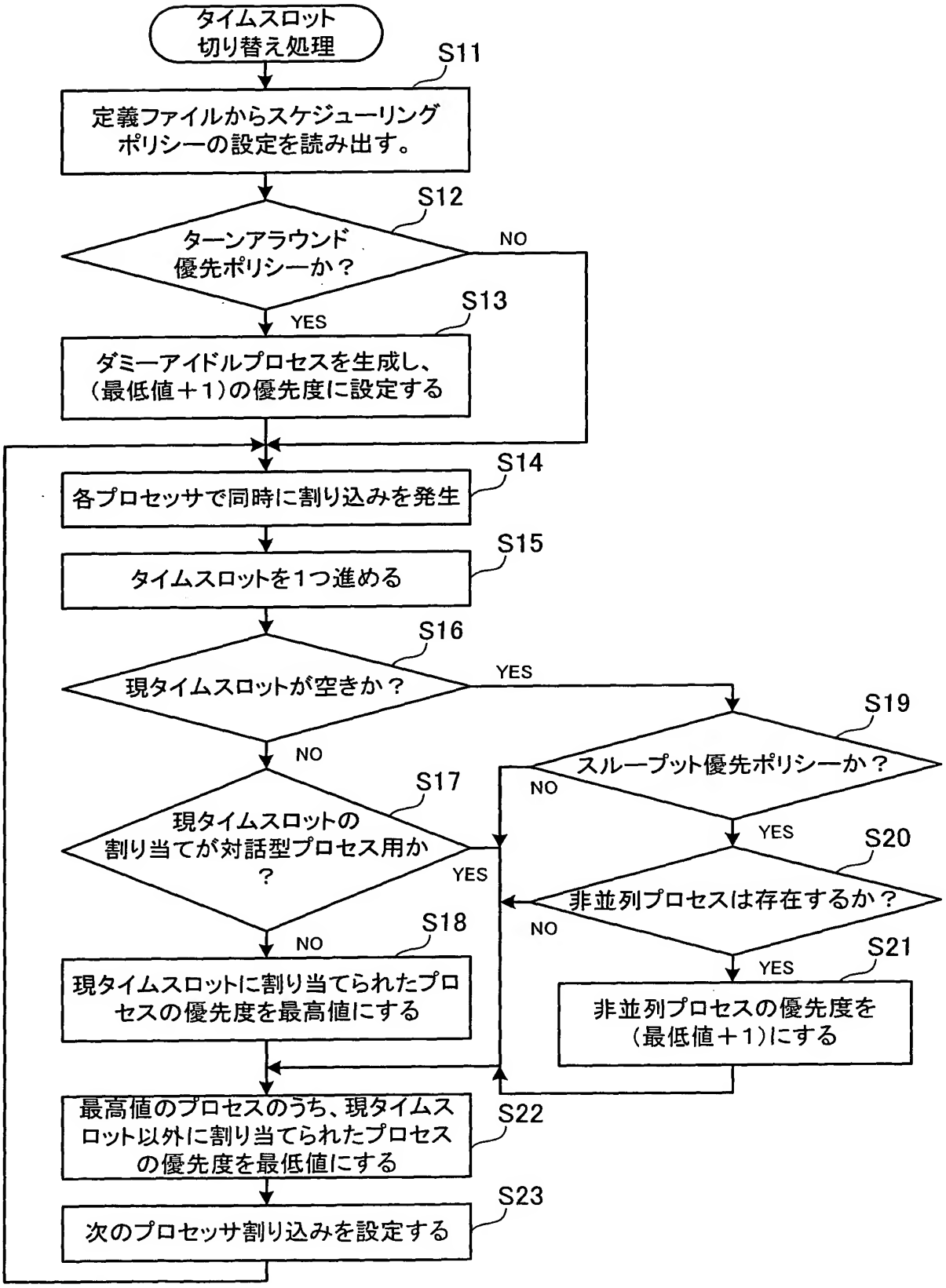

タイムスロット
切り替え処理
S11
定義ファイルからスケジューリング
ポリシーの設定を読み出す。
S12
ターンアラウンド
優先ポリシーか？
NO
YES
S13
ダミーアイドルプロセスを生成し、
（最低値＋1）の優先度に設定する
S14
各プロセッサで同時に割り込みを発生
S15
タイムスロットを1つ進める
S16
現タイムスロットが空きか？
YES
NO
S19
スループット優先ポリシーか？
NO
YES
S17
現タイムスロットの
割り当てが対話型プロセス用か
？
YES
S20
非並列プロセスは存在するか？
NO
YES
S18
NO
現タイムスロットに割り当てられたプロ
セスの優先度を最高値にする
S21
非並列プロセスの優先度を
（最低値＋1）にする
最高値のプロセスのうち、現タイムス
ロット以外に割り当てられたプロセス
の優先度を最低値にする
S22
次のプロセッサ割り込みを設定する
S23

図9

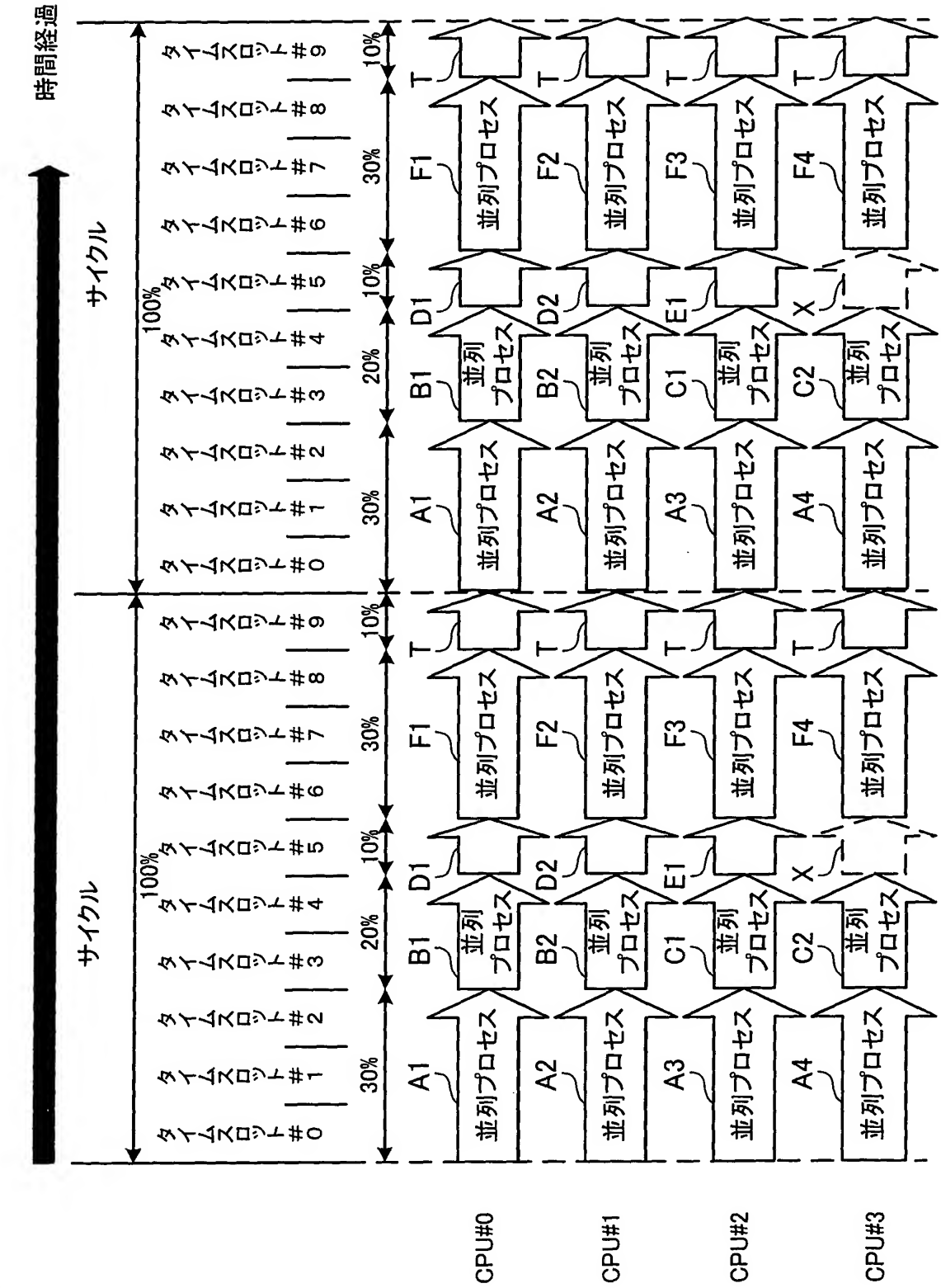

時間経過
サイクル
サイクル
100%
100%
30%
20%
10%
30%
10%
30%
20%
10%
タイムスロット#0
タイムスロット#1
タイムスロット#2
タイムスロット#3
タイムスロット#4
タイムスロット#5
タイムスロット#6
タイムスロット#7
タイムスロット#8
タイムスロット#9
A1
A2
A3
A4
B1
B2
C1
C2
D1
D2
E1
X
F1
F2
F3
F4
T
並列プロセス
CPU#0
CPU#1
CPU#2
CPU#3

図10

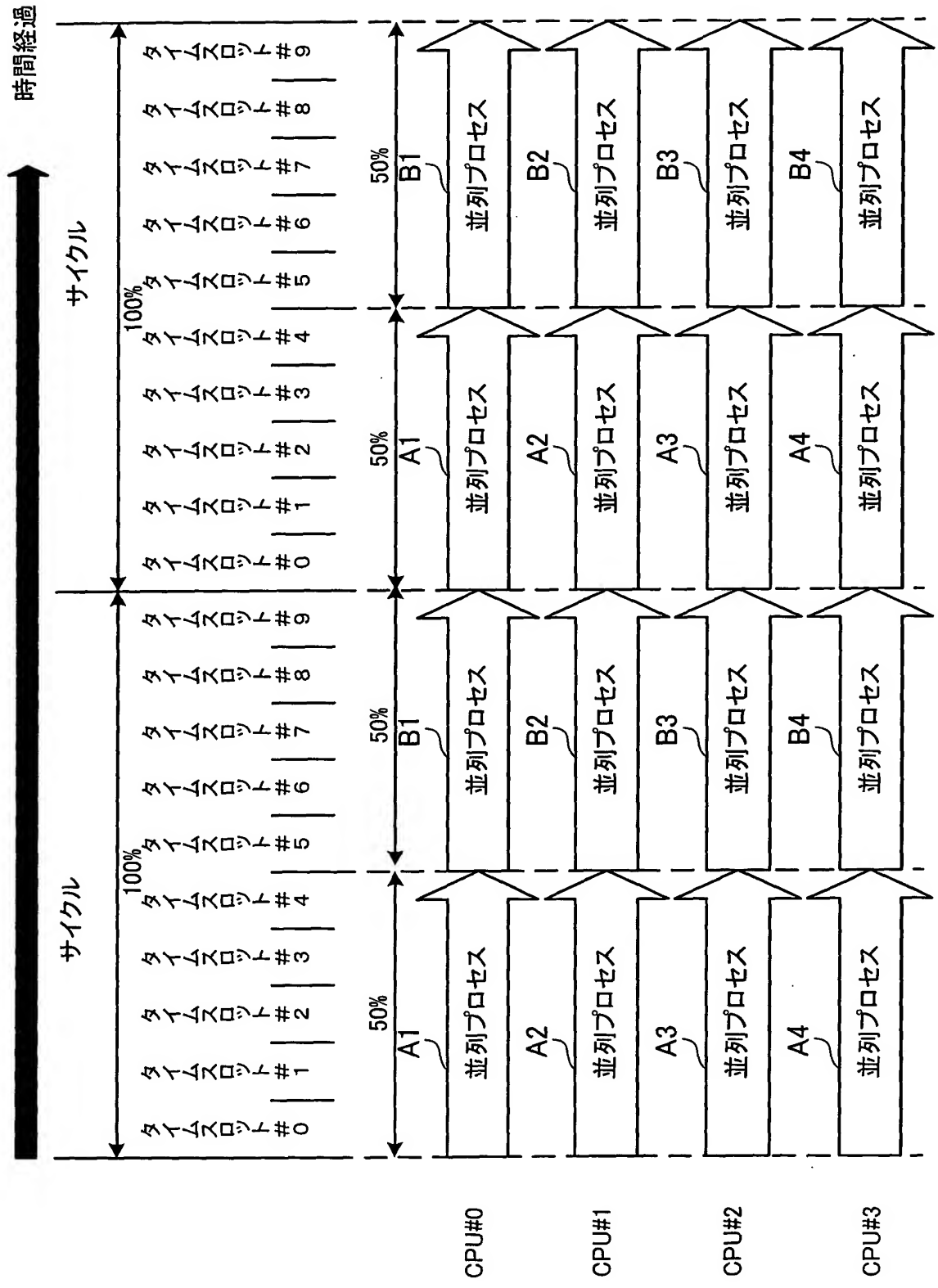

時間経過
サイクル
サイクル
100%
50%
タイムスロット#0
タイムスロット#1
タイムスロット#2
タイムスロット#3
タイムスロット#4
タイムスロット#5
タイムスロット#6
タイムスロット#7
タイムスロット#8
タイムスロット#9
A1
A2
A3
A4
B1
B2
B3
B4
並列プロセス
CPU#0
CPU#1
CPU#2
CPU#3

図１１

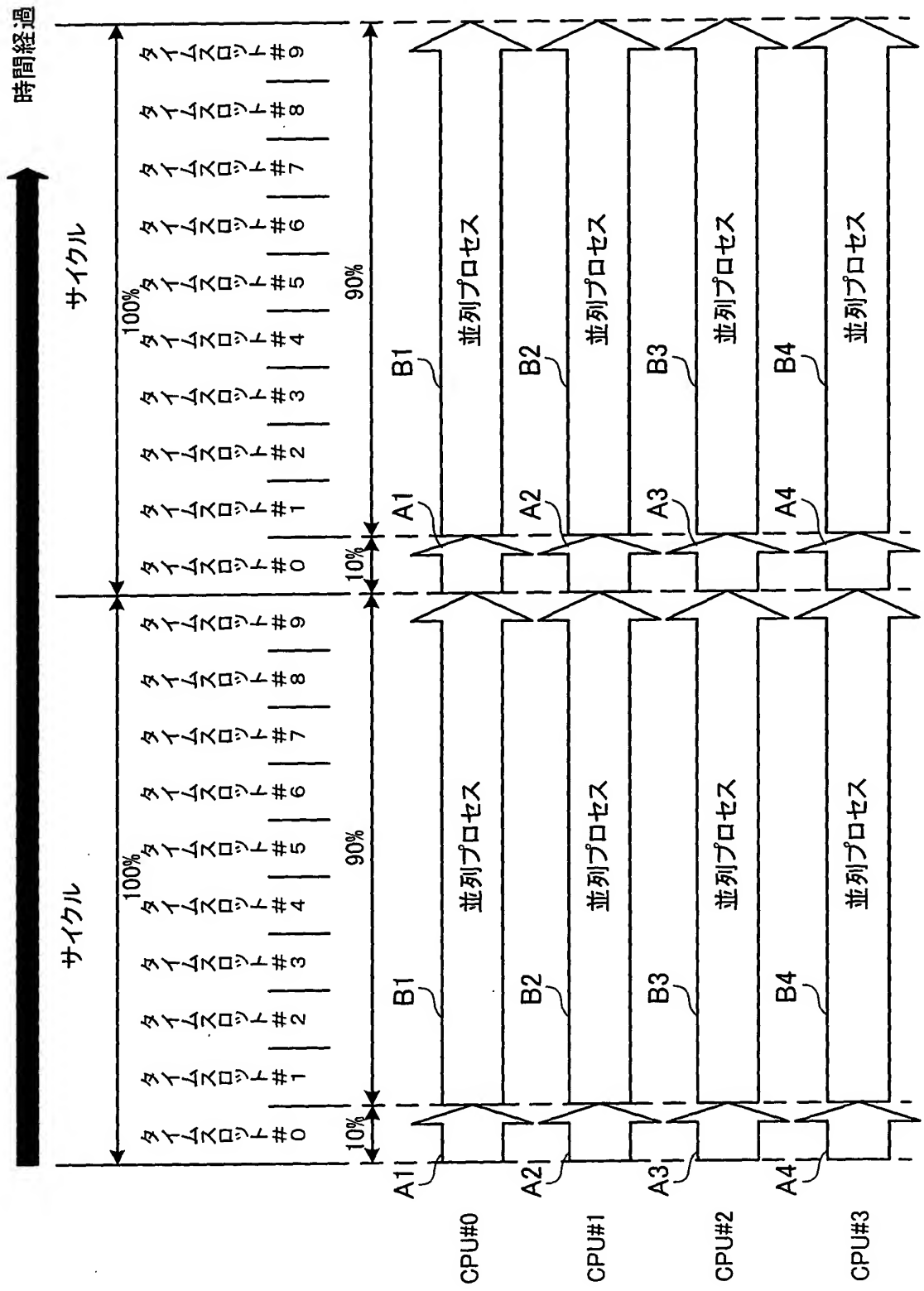
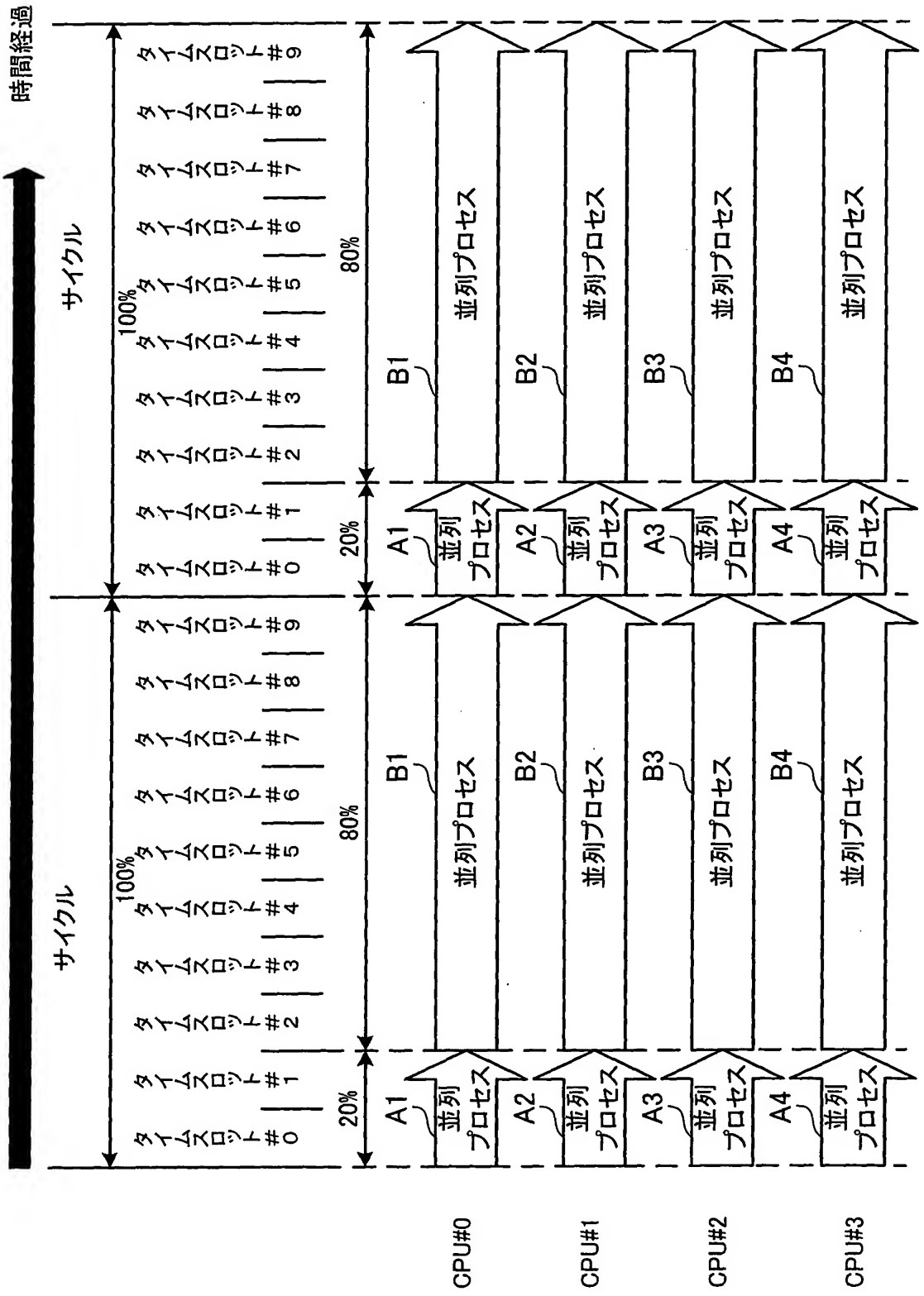

時間経過
サイクル
サイクル
100%
90%
10%
タイムスロット#0
タイムスロット#1
タイムスロット#2
タイムスロット#3
タイムスロット#4
タイムスロット#5
タイムスロット#6
タイムスロット#7
タイムスロット#8
タイムスロット#9
A1
A2
A3
A4
B1
B2
B3
B4
並列プロセス
CPU#0
CPU#1
CPU#2
CPU#3

図12


時間経過
サイクル
サイクル
100%
80%
20%
タイムスロット#0
タイムスロット#1
タイムスロット#2
タイムスロット#3
タイムスロット#4
タイムスロット#5
タイムスロット#6
タイムスロット#7
タイムスロット#8
タイムスロット#9
A1
A2
A3
A4
B1
B2
B3
B4
並列プロセス
並列プロセス
CPU#0
CPU#1
CPU#2
CPU#3

スケジューリングポリシー：スループット優先ポリシー

図13

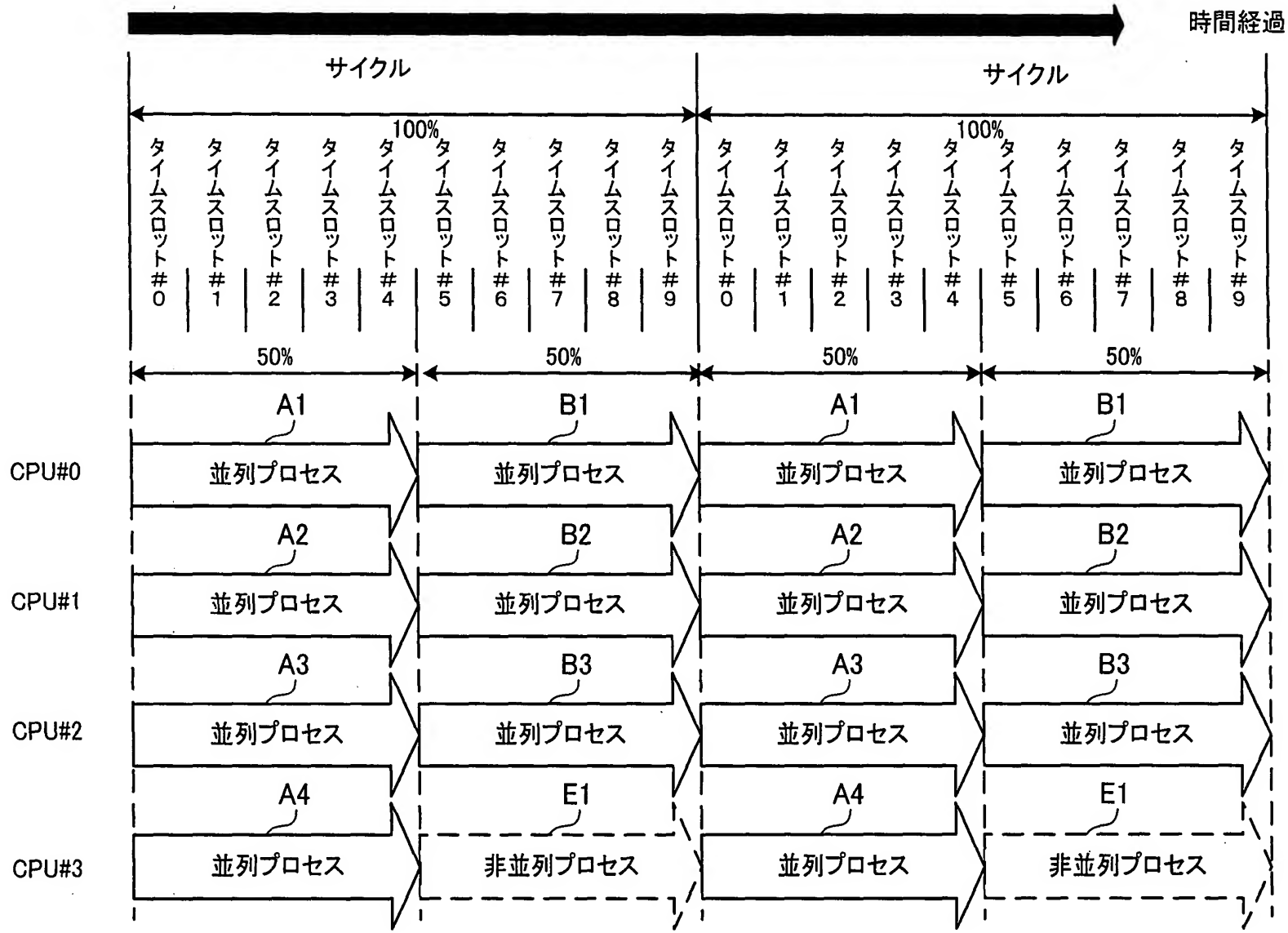


図14

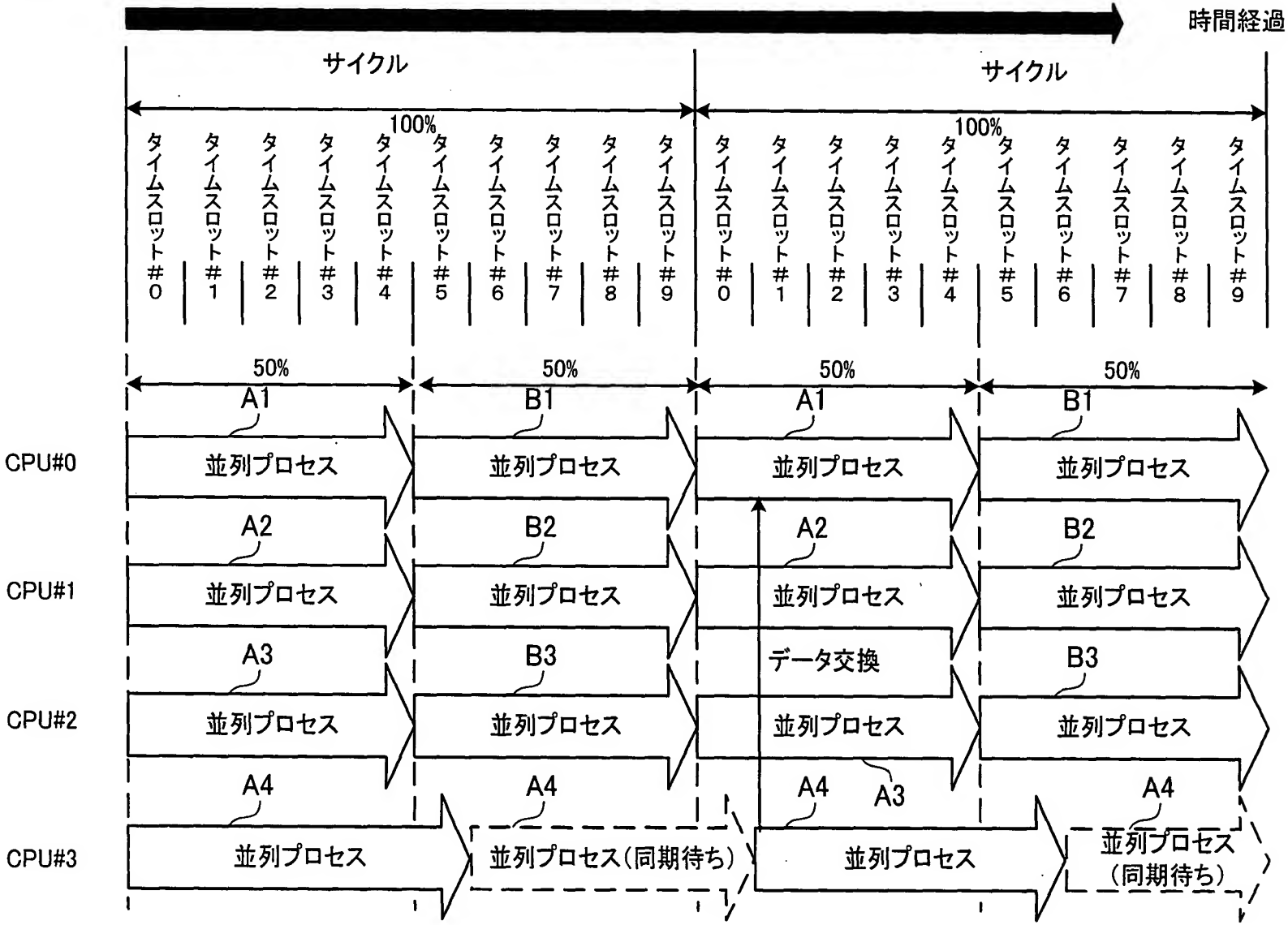
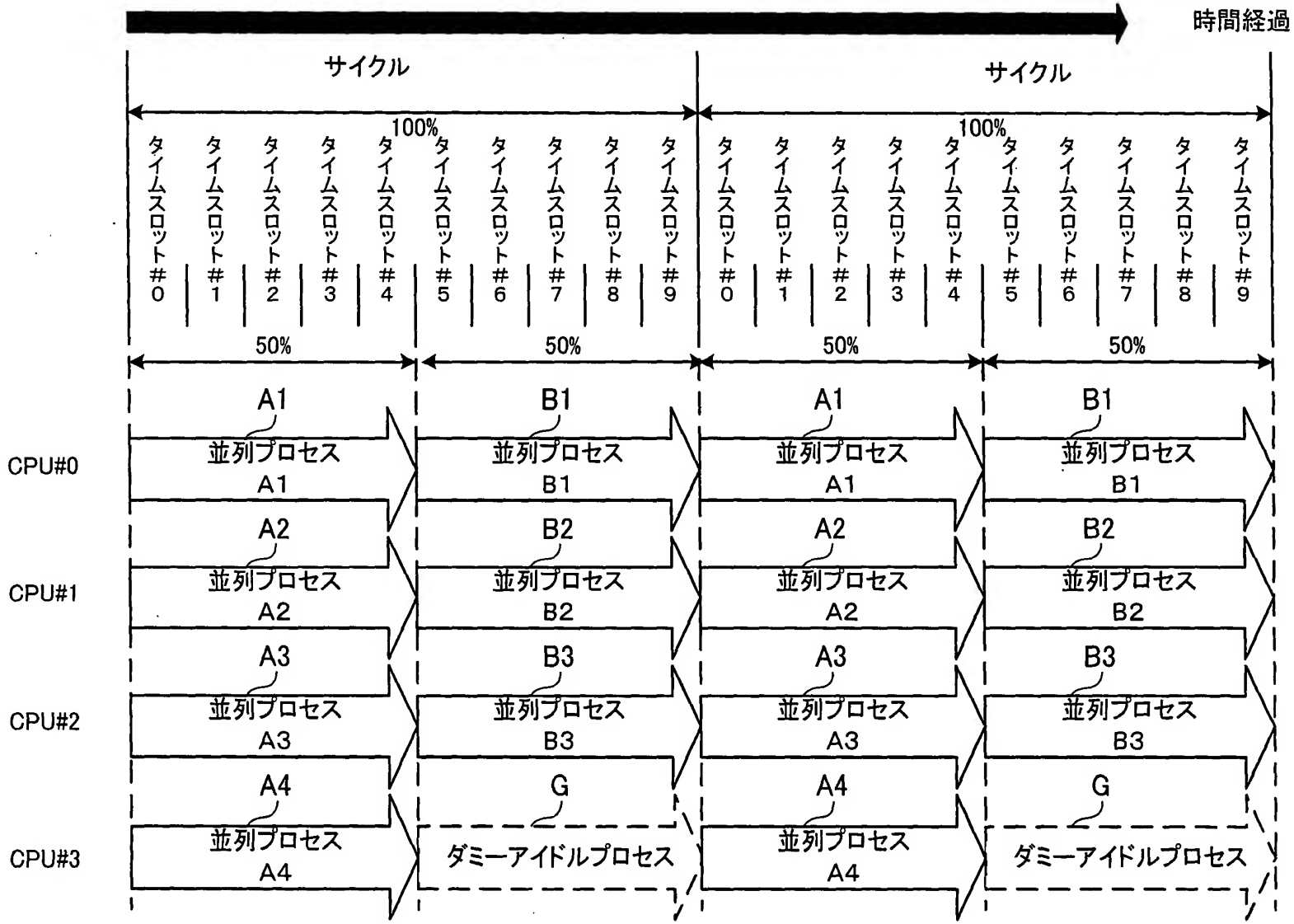

スケジューリングポリシー：スループット優先ポリシー（非並列プロセス、対話型プロセス無し）
時間経過
サイクル
サイクル
100%
100%
タイムスロット#0
タイムスロット#1
タイムスロット#2
タイムスロット#3
タイムスロット#4
タイムスロット#5
タイムスロット#6
タイムスロット#7
タイムスロット#8
タイムスロット#9
50%
50%
50%
50%
A1
B1
A1
B1
CPU#0
並列プロセス
並列プロセス
並列プロセス
並列プロセス
A2
B2
A2
B2
CPU#1
並列プロセス
並列プロセス
並列プロセス
並列プロセス
A3
B3
データ交換
B3
CPU#2
並列プロセス
並列プロセス
並列プロセス
並列プロセス
A4
A4
A4
A3
A4
CPU#3
並列プロセス
並列プロセス（同期待ち）
並列プロセス
並列プロセス
（同期待ち）

図15

スケジューリングポリシー:ターンアラウンド優先ポリシー



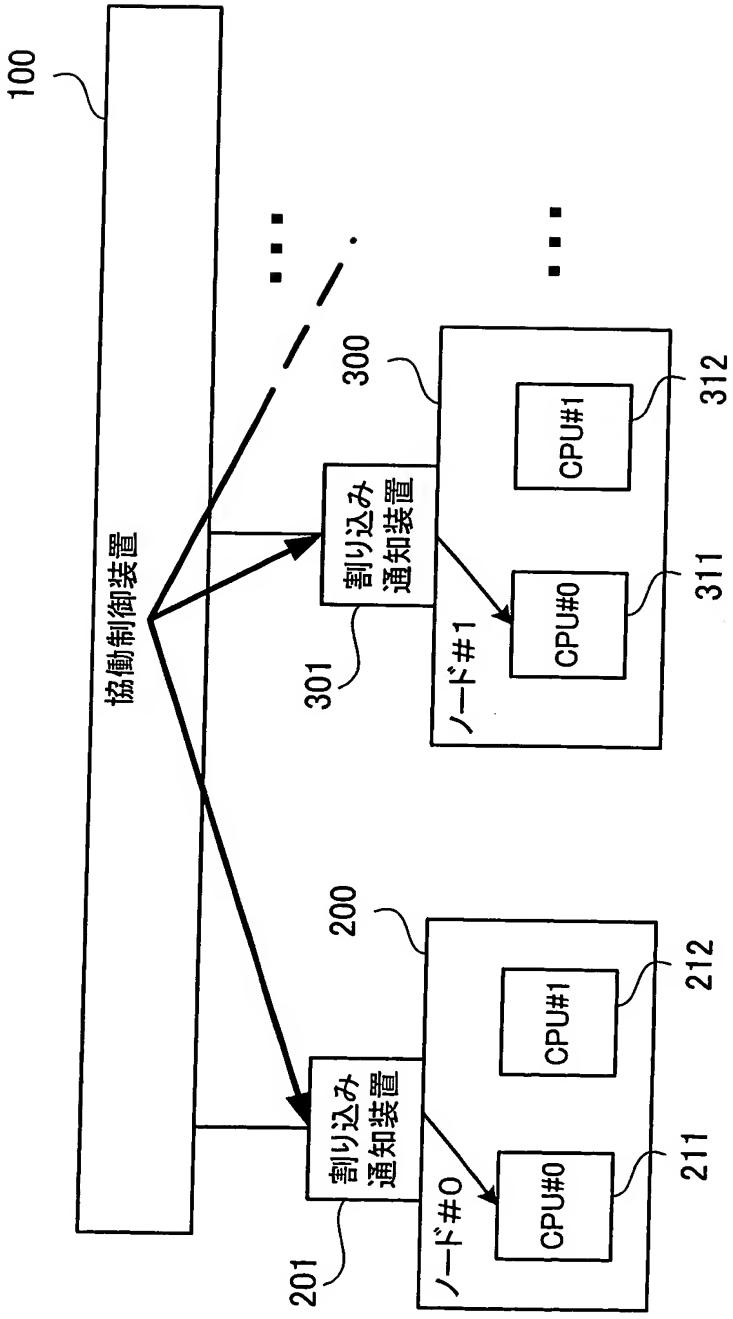

100
協働制御装置
200
201
割り込み
通知装置
ノード#0
CPU#0
211
CPU#1
212
300
301
割り込み
通知装置
ノード#1
CPU#0
311
CPU#1
312

図16

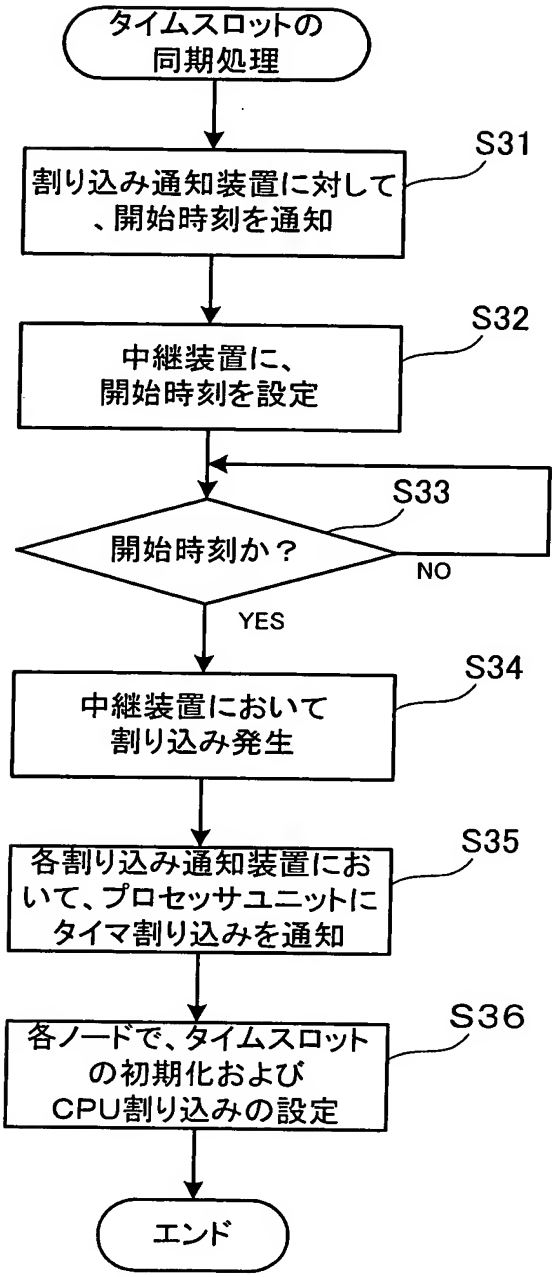

タイムスロットの
同期処理
S31
割り込み通知装置に対して
、開始時刻を通知
S32
中継装置に、
開始時刻を設定
S33
開始時刻か？
NO
YES
S34
中継装置において
割り込み発生
S35
各割り込み通知装置にお
いて、プロセッサユニット
タイマ割り込みを通知
S36
各ノードで、タイムスロット
の初期化および
CPU割り込みの設定
エンド

図17

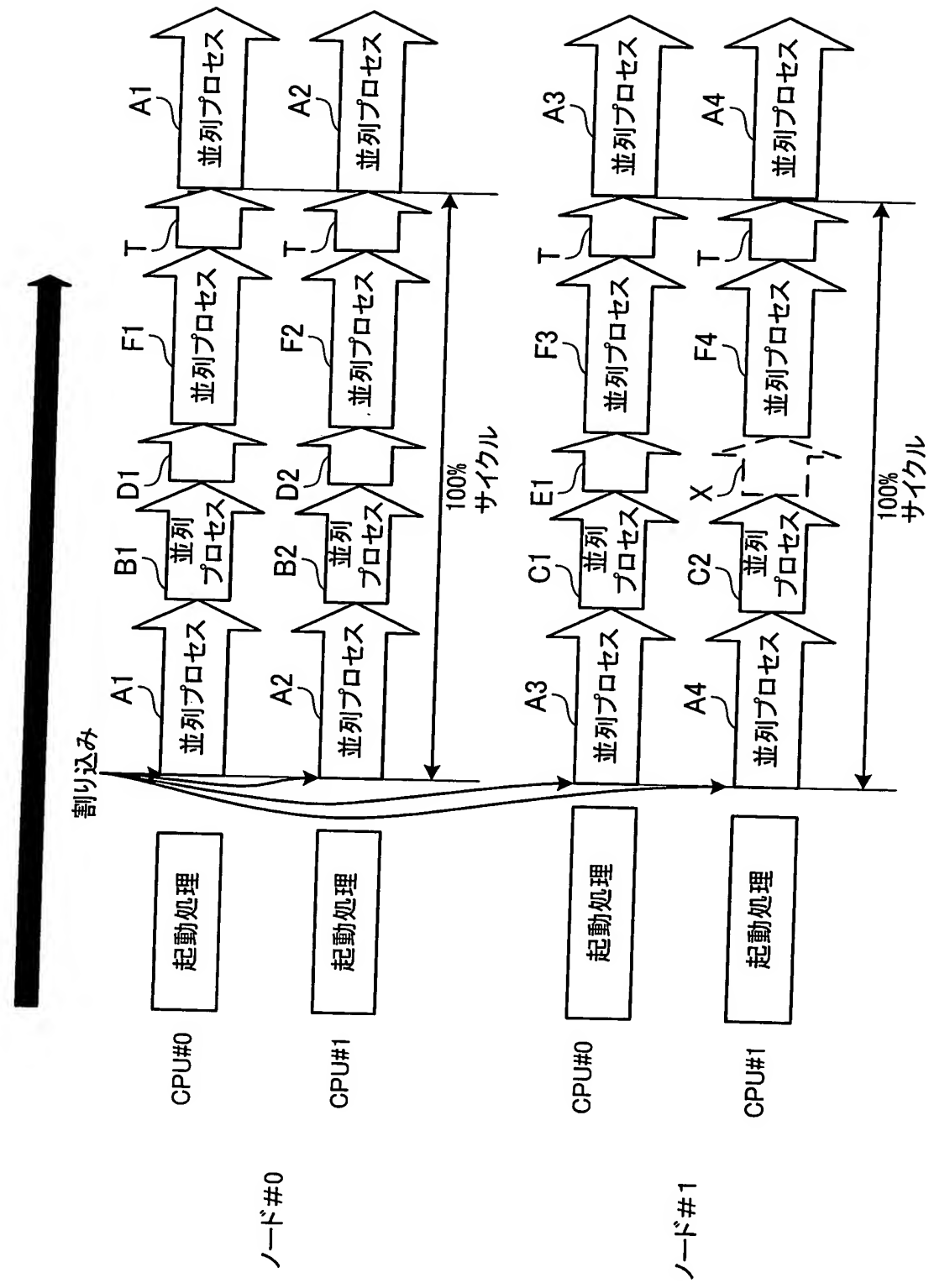

ノード#0
ノード#1
CPU#0
CPU#1
CPU#0
CPU#1
起動処理
起動処理
起動処理
起動処理
割り込み
A1
並列プロセス
A2
並列プロセス
A3
並列プロセス
A4
並列プロセス
B1
並列プロセス
B2
並列プロセス
C1
並列プロセス
C2
並列プロセス
D1
D2
E1
X
F1
並列プロセス
F2
並列プロセス
F3
並列プロセス
F4
並列プロセス
T
T
T
T
A1
並列プロセス
A2
並列プロセス
A3
並列プロセス
A4
並列プロセス
100%
サイクル
100%
サイクル

図18